Quantum Error Correction for the Toric Code

Master Thesis
in Theoretical Physics

at the University of Bern
Institute for Theoretical Physics

by Matti Zbinden

Supervisor: Prof. Uwe-Jens Wiese

Bern, February 12, 2019

Abstract

In this thesis, the toric code error correction process, one method to check a quantum memory storage for errors and correcting them, is analysed theoretically. The focus was set on the determination of the accuracy threshold, the upper limit p_{crit} of the probability with which the manifestation of errors can be allowed during consecutive measurements for errors, so that the non-erroneous state of the quantum memory storage can be restored. To do this, an analogy between the toric code error correction model (TCECM) and the random-bond Ising model (RBIM) was used. In this analogy, the probability of error creation p_{EC} in the TCECM is equivalent to the probability to have an antiferromagnetic bond p_{AFB} in the RBIM. The behaviour of the TCECM with increasing p_{EC} corresponds to the behaviour of the RBIM along the so called Nishimori line with increasing p_{AFB}. The two phases of the TCECM, where the non-erroneous state can be restored respectively cannot, correspond to the ferromagnetic respectively paramagnetic phase of the RBIM. In the RBIM, the transition between these two phases along the Nishimori line occurs at the so called Nishimori point. Thus, by numerically simulating the RBIM along this Nishimori line and determining the position of the phase transition, the accuracy threshold p_{crit} of the TCECM can be determined.

The two different variants of the algorithm that were used yielded $p_{crit}=0.1081\pm0.0007$ respectively $p_{crit}=0.108\pm0.001$ as a result for this accuracy threshold. These values deviate slightly from reference values found in the literature. The reason for this remains somewhat unclear, but likely sources were determined and propositions made on how to get rid of them by improving the method.

Table of content

1. Introduction

Currently, quantum computers are envisioned as the next great revolution in computer development. They allow the performance of a new, broader class of operations on the information that is put in. This in turn opens up new ways to solve mathematical and physical problems. Such quantum algorithms will have their largest impact on problems which previously could only be solved with a large expenditure of time and/or storage space, but for which these new operations provide more efficient approaches. This makes quantum computation an important emerging field of study.

However, in order to actually build a quantum computer that is able to reliably perform calculations, many different technical problems need to be solved. One of these problems is to ensure that external influences do not cause errors in the stored data during calculation. One way to address this problem is through the use of so called *surface codes*. These are described, after an introduction into quantum computation, in Section 2. The focus of this description is set on the theoretical model behind the methods used for the recognition of errors in the stored data and their correction for a specific surface code, named the *toric code*.

The objective of this thesis is to determine the so called *accuracy threshold*, the upper bound for the probability for errors to occur between different rounds of error recognition that limits the ability of the system to restore the non-erroneous state of the stored data. This is done using the analogy of the problem to the two-dimensional random-bond Ising model. The calculations are performed numerically by applying worm algorithms. To test the correctness of the results, two different types of worm algorithms are used for each calculation. The method is described in more detail in Section 3, explaining how the algorithms were first tested on the two-dimensional standard Ising model before being applied to the two-dimensional random-bond Ising model. The results of the tests in the standard Ising model are then presented and discussed in Section 4. They are followed by the presentation and discussion of the results for the accuracy threshold from the simulation of the random-bond Ising model in Section 5. Thereafter, the performance of the two types of worm algorithms is compared in Section 6. Then, further findings from the simulations of these Ising models are presented in Section 7. Finally, conclusions are drawn in Section 8.

2. Theoretical background

This section aims at providing an overview about quantum computation in general and the realisation of a quantum computer using so called *surface codes* in particular. Then the focus is laid on a particular form of surface code, the *toric code*. For this, the error and correction model is presented and discussed in detail, just like their relation to the random-bond Ising model. This provides the necessary theoretical background to understand the calculations performed in the following sections as well as to understand the results.

2.1. Quantum computers

In order to understand what a quantum computer is, it is useful to first understand how exactly a "classical" computer performs a calculation. A classical computer can be considered as an electronic experimental set-up that is analogous to a certain way to solve a mathematical or physical problem (a so called *algorithm*), respectively can be put into a configuration that has this analogy. It consists of a sequence of so called *classical gates*, which, obeying the laws of classical electrodynamics, in this set-up perform operations analogous to the logical operations of the algorithm. On the set-up an initial configuration is set, which corresponds to the specifications of the problem. Then, the experiment is performed. This mostly consists of a temporal evolution of the set-up that depends on the initial configuration. Eventually, the final configuration is determined. Using the analogy to the algorithm, out of this final configuration the solution of the problem is then interpreted. A quantum computer works basically the same way. Just instead of making use of the laws of classical electrodynamics, it is based on the laws of quantum physics. This leads to some essential differences between these two types of computers and how they work, e.g. in the use of so called *quantum gates* instead of the classical gates. These differences are discussed in the following subsections and are based on Nielsen & Chuang [1].

2.1.1. Information storage

In classical computers, information is stored in *bits*, represented by either the presence or the absence of an electric charge respectively current. Consequently, there are two absolute and contrary (classical) states of which just the one or the other, but not both, can be present at a time. In quantum computers on the other hand, the information is stored in so called *quantum bits* or *qubits*. These are realized by a quantum physical system that can be in two (quantum) states, $|0\rangle$ and $|1\rangle$, or any superposition of them, $\alpha|0\rangle+\beta|1\rangle$, which satisfies $|\alpha|^2+|\beta|^2=1$. The strict exclusion constraint found in classical computers does not apply here any longer. This increases the number of different states that are realizable to infinity. However, when a quantum state is measured at the end of the calculation, it will collapse to one of the basis states of the corresponding measurement operators. For the initial basis introduced above this would be either $|0\rangle$ or $|1\rangle$. The probability for this is given by $|\alpha|^2$ respectively $|\beta|^2$. Since the solution to the mathematical respectively physical problem is interpreted from the probability distribution with which the different measurement outcomes of the finally applied measurement operator manifest, a calculation on a quantum computer in general needs to be performed a large number of times to determine a good estimate for this distribution.

2.1.2. Information processing

In classical computers, information is processed by inserting one or multiple bits into a so called *gate*. A new bit then exits the gate. Its state depends on the states of the inserted bits and the type of the gate. The number of different gates is determined by the number of different possible inputs,

which is 2^n with n the number of bits that is put in. For each of these inputs, the gate determines a specific output. Each type of gate differs in this assignment. Hence there are $2^{(2^n)}$ different gates. However, not all of these gates need to be realized physically in a computer. Most of them can be built as a combination of other gates. This largely reduces the number of gates that actually need to be realized to a small set. Such a set is called *functionally complete*. A specific set of gates out of which by building the right combination each other type of gate can be created is referred to as *universal*.
In quantum computers, on the other hand, the information processing has to abide to the rules of quantum physics. This means that a quantum gate takes the form of a unitary operator that is applied on the inserted quantum state to form the output state. Consequently, the number of possible quantum gates is infinite. However, like for the classical gates, there are functionally complete sets of quantum gate types that allow the construction of any other type. One of these consists of the two qubit *controlled NOT-gate* (CNOT) together with the single qubit gates. The way the CNOT-gate works is illustrated in Figure 2.1.1.

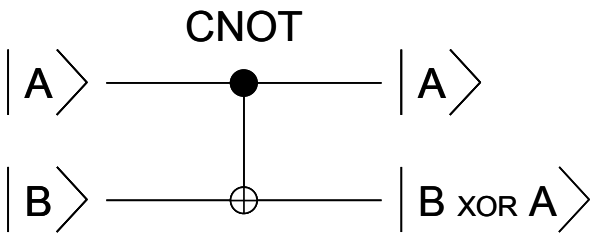

Figure 2.1.1: Structure of the controlled NOT-gate. In this quantum gate two qubit states, $|\mathrm{A}\rangle$ *and* $|\mathrm{B}\rangle$ *, are inserted. Each of them is a quantum state that has the form* $\alpha_i|0\rangle+\beta_i|1\rangle$ *. The total inserted quantum state is thus given by the tensor product of these two ket-vectors. It can be expressed as* $a|0_\mathrm{A}\,0_\mathrm{B}\rangle+b|0_\mathrm{A}\,1_\mathrm{B}\rangle+c|1_\mathrm{A}\,0_\mathrm{B}\rangle+d|1_\mathrm{A}\,1_\mathrm{B}\rangle$ *, where the index indicates to which qubit a certain basis state belongs. The output of this quantum gate is such that the state of the first qubit,* $|\mathrm{A}\rangle$ *, remains unchanged while that of the second one,* $|\mathrm{B}\rangle$ *, gets transformed into* $|\mathrm{B}\,\mathrm{XOR}\,\mathrm{A}\rangle$ *, where XOR denotes the logical operation known as* exclusive OR*. This logical operation has two inputs, A and B, which can be either true (1) or false (0). Its output is true (1), if the inputs are different (A=1, B=0 respectively A=0, B=1), and false (0), if the inputs are the same (A=0, B=0 respectively A=1, B=1). This means that in the CNOT-gate the prefactor in front of the basis ket-vector* $|i_\mathrm{A}, j_\mathrm{B}\rangle$ *gets shifted in front of the basis ket-vector* $|i_\mathrm{A}, j_\mathrm{B}\,\mathrm{XOR}\,i_\mathrm{A}\rangle$ *. Thus the output of the CNOT-gate is given by* $a|0_\mathrm{A}\,0_\mathrm{B}\rangle+b|0_\mathrm{A}\,1_\mathrm{B}\rangle+d|1_\mathrm{A}\,0_\mathrm{B}\rangle+c|1_\mathrm{A}\,1_\mathrm{B}\rangle$ *.*

2.1.3. Information copying and deletion

In classical computers the information stored in bits can easily be copied or deleted. However, since in quantum computers the quantum gates can only be unitary operators, they are always reversible. Therefore, a quantum state can be neither deleted nor copied, because both processes contain the loss of quantum information (that of the information-bearing qubit itself respectively that of the qubit on which the information should be copied). Such a loss, however, is irreversible. This is because in the deletion the initial quantum state can be arbitrary, the final state, however, is given. Consequently, a reverse transformation should be able to map a given state onto any arbitrary state. This cannot be achieved by a unitary transformation. The proof for copying goes mutatis mutandis.

2.2. Surface codes

While the abstract theoretical description of quantum computers is well-established and works quite nicely, their actual development turns out to face some technical challenges. These problems first need to be solved in order to build a quantum computer that is able to reliably perform a calculation. One of the problems is how to prevent the modification of the information stored in the quantum

system by perturbations from external sources, so that a calculation can be performed correctly. One way to address this problem is by using so called *surface codes*. Their discussion in the next subsections follows E. Dennis et al. [2].

2.2.1. Structure of surface codes

In surface codes, instead of storing a certain piece of information in just a single qubit, a large set of qubits is used. Each of these qubits corresponds to a spin ½ quantum system, so that it can either be in the state *spin up*, *spin down* or in a superposition of these two states. The qubits are arranged on the links of a two-dimensional lattice surface as shown in Figure 2.2.1. Thus, if the lattice has a horizontal length of L_x links and a vertical length of L_y links, it consists of a total of $2L_xL_y$ links. Hence, it gives rise to a $2^{2L_xL_y}$ -dimensional Hilbert space. At each site of the lattice, four qubits connect. These neighbouring qubits are able to interact with each other. These interactions as well as the geometry of the surface are chosen in a way that the ground state is degenerate. This means that there are several distinct quantum states of the joint system that have the same energy, which is also the minimal energy the system can realize. These ground quantum states of the joint system are in turn used as the basis states of qubits. These qubits are the ones that are used for the storage of information and the computation. They are thus referred to as *logical qubits* while those qubits forming the lattice are called *physical qubits*.

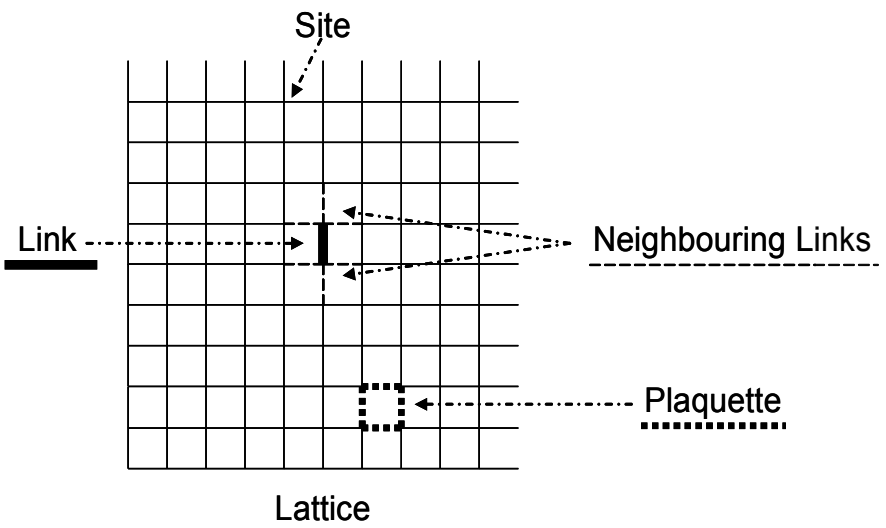

Figure 2.2.1: Structure of the surface code lattice. Each point on which two lines cross is called a site*. The horizontal or vertical lines connecting two adjacent sites are referred to as* links*. One link is highlighted by being drawn broader. The six links neighbouring this one are highlighted by being drawn dashed. A square formed by four connected links is called a* plaquette*. One such plaquette is highlighted by being drawn broader and dotted. The complete structure is known as a* lattice*.*
The dashed arrows are not part of the lattice but placed to point at different structural parts.

The spin ½ nature of the physical qubits can be described with the Pauli matrices σ_ℓ^a where ℓ determines the position of the link on which the Pauli matrix acts and the σ^a , $a \in \{1,2,3\}$ are given by

$$\sigma^1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \ , \ \sigma^2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \ , \ \sigma^3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \ . \tag{2.2.1}$$

They obey the commutation relation

$$[\sigma_\ell^a, \sigma_{\ell'}^b] = 2i\,\delta_{\ell\,\ell'}\epsilon^{abc}\sigma_\ell^c \tag{2.2.2}$$

where δ_{xy} is the Kronecker delta and ϵ^{ijk} is the Levi-Civita tensor. Furthermore, the Pauli matrices are Hermitian and their own inverse.

2.2.2. Toric code

One specific type of surface code is the toric code. Here each pair of opposite ends of the lattice are joined. Hence, the lattice takes the form of the surface of a torus. Consequently, it no longer has a boundary. The interactions between the links is such that the Hamiltonian H of the system is given by

$$\begin{aligned} H &= -\sum_{i\in\text{Sites}} J_i^{+}\cdot X_{+,i} \;-\sum_{j\in\text{Plaquettes}} J_j^{\square}\cdot Z_{\square,j} \\ &= -\sum_{i\in\text{Sites}} J_i^{+} \prod_{\ell\in \substack{\text{links adjacent}\\ \text{to the site } i}} \sigma_\ell^1 \;-\sum_{j\in\text{Plaquettes}} J_j^{\square} \prod_{\ell'\in \substack{\text{links in the}\\ \text{plaquette } j}} \sigma_{\ell'}^3, \end{aligned} \tag{2.2.3}$$

where the first sum goes over all sites of the lattice, the second sum over all of its plaquettes (see Figure 2.2.1 for the definition of the plaquette). $J_i^{+}, J_j^{\square}\in\mathbb{R}$ are the coupling constants. They are set to be

$$J_i^{+}=J_j^{\square}=J=const.>0 \quad \forall\, i,\, j\in\text{links}. \tag{2.2.4}$$

$X_{+,i}$ refers to the X-star operator at the site i, it applies the Pauli-X operator σ^1 on the four links adjacent to that site (see also Figure 2.2.2)

$$X_{+,i}=\prod_{\ell\in \substack{\text{links adjacent}\\ \text{to the site } i}} \sigma_\ell^1. \tag{2.2.5}$$

$Z_{\square,j}$ refers to the Z-plaquette operator at the plaquette j, it applies the Pauli-Z operator σ^3 on the four links forming that plaquette (see also Figure 2.2.2)

$$Z_{\square,j}=\prod_{\ell\in \substack{\text{links in the}\\ \text{plaquette } j}} \sigma_\ell^3. \tag{2.2.6}$$

Like the Pauli matrices they are made of, the X-star and Z-plaquette operators are unitary and their own inverse. Furthermore, all the X-star and Z-plaquette operators commute with one another and thus also with the Hamiltonian H, even though σ^1 and σ^3 do not commute when acting on the same physical qubit.

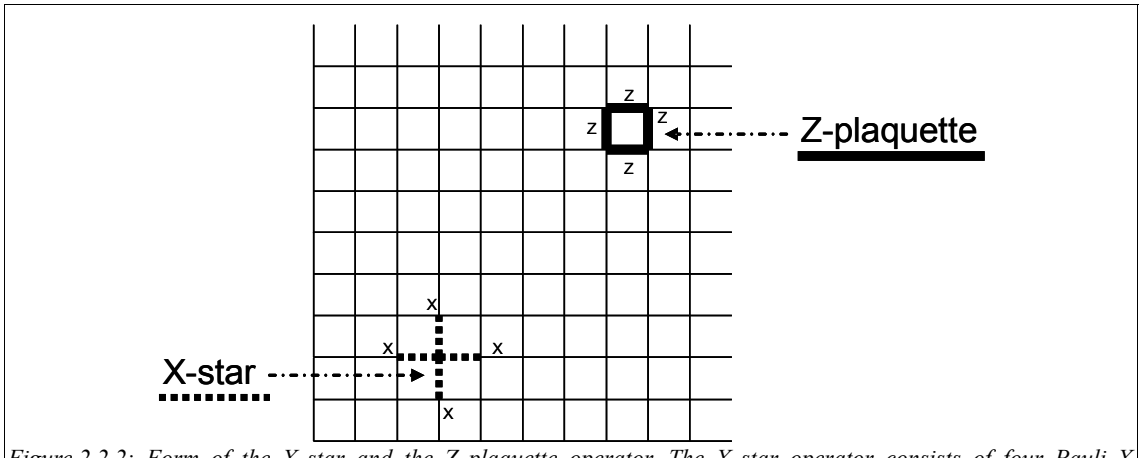

Figure 2.2.2: Form of the X-star and the Z-plaquette operator. The X-star operator consists of four Pauli-X operators applied on the links adjacent to a site. The Z-plaquette operator consists of four Pauli-Z operators applied to the links forming a plaquette.

The reason for this is that any pair of an X-star and a Z-plaquette operator either overlap at none or

two qubits. When they do not overlap, there is no qubit on which both a σ^1 and a σ^3 operator act. Instead these Pauli operators are preceded respectively followed by identity operators with which they trivially commute. The proof for the other case, where these two operators overlap at two qubits, is a bit more sophisticated:

σ^1 and σ^3 do not commute. If they are applied on a certain quantum state, the final result depends on the order in which they have been applied.

$$\begin{aligned}\sigma^1\sigma^3\begin{pmatrix}\alpha\\ \beta\end{pmatrix}&=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ -\beta\end{pmatrix}=\begin{pmatrix}-\beta\\ \alpha\end{pmatrix},\\ \sigma^3\sigma^1\begin{pmatrix}\alpha\\ \beta\end{pmatrix}&=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\beta\\ \alpha\end{pmatrix}=\begin{pmatrix}\beta\\ -\alpha\end{pmatrix}=(-1)\cdot\begin{pmatrix}-\beta\\ \alpha\end{pmatrix}\end{aligned} \tag{2.2.7}$$

The difference in the final state between the two orders of application is, however, just a factor of -1. Hence, when these two Pauli operators are applied on two quantum states, the difference between the two orders in which they are applied becomes $(-1)^2=1$. This occurs because of the bilinearity of the tensor product $\otimes$ which combines the two quantum states.

$$\begin{aligned}(\sigma^1\otimes\sigma^1)(\sigma^3\otimes\sigma^3)\left(\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\otimes\begin{pmatrix}\gamma\\ \delta\end{pmatrix}\right) &= (\sigma^1\sigma^3\begin{pmatrix}\alpha\\ \beta\end{pmatrix})\otimes(\sigma^1\sigma^3\begin{pmatrix}\gamma\\ \delta\end{pmatrix})=\begin{pmatrix}-\beta\\ \alpha\end{pmatrix}\otimes\begin{pmatrix}-\delta\\ \gamma\end{pmatrix},\\ (\sigma^3\otimes\sigma^3)(\sigma^1\otimes\sigma^1)\left(\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\otimes\begin{pmatrix}\gamma\\ \delta\end{pmatrix}\right) &= (\sigma^3\sigma^1\begin{pmatrix}\alpha\\ \beta\end{pmatrix})\otimes(\sigma^3\sigma^1\begin{pmatrix}\gamma\\ \delta\end{pmatrix})=\begin{pmatrix}\beta\\ -\alpha\end{pmatrix}\otimes\begin{pmatrix}\delta\\ -\gamma\end{pmatrix}\\ &=\left((-1)\cdot\begin{pmatrix}-\beta\\ \alpha\end{pmatrix}\right)\otimes\left((-1)\cdot\begin{pmatrix}-\delta\\ \gamma\end{pmatrix}\right)=(-1)^2\cdot\left(\begin{pmatrix}-\beta\\ \alpha\end{pmatrix}\otimes\begin{pmatrix}-\delta\\ \gamma\end{pmatrix}\right)\\ &= \begin{pmatrix}-\beta\\ \alpha\end{pmatrix}\otimes\begin{pmatrix}-\delta\\ \gamma\end{pmatrix}\end{aligned} \tag{2.2.8}$$

As a consequence, the final state after the application of these two Pauli operators on two quantum states is the same, no matter which one was applied first. Thus, the two operators $\sigma^1\otimes\sigma^1$ and $\sigma^3\otimes\sigma^3$ commute. Therefore also the X-star and Z-plaquette operators commute if they overlap on two physical qubits.

Since two σ^1 operators that are applied to the same physical qubit trivially commute with each other, as do two σ^3 operators, the different X-star operators commute among themselves, just like the Z-plaquette operators. Furthermore, since the Hamiltonian only consists of a sum of these operators, each of them also commutes with the Hamiltonian.

The toric code has two $\mathbb{Z}_2$ gauge symmetries, one of the actual lattice and the other of the dual lattice. For the error model and its correction model described in Section 2.2.3 respectively Section 2.2.4 only the latter one is of importance. Therefore, only the $\mathbb{Z}_2$ gauge symmetry of the dual lattice will be discussed here. This gauge symmetry demands that the application of any of the Z-plaquette operators $Z_{\square,j}$ or of a combination of them on the quantum state of the lattice must leave that state invariant.

The determination of the ground state of the toric code is quite elaborate:

As shown in Section 2.2.2, each of the $2L_xL_y$ physical qubits of the lattice has two spin states. This yields $2^{2L_xL_y}$ different configurations of these spin states that can be realized on the lattice. Since the assignment of the labels *spin up* and *spin down* to the two quantum states of a physical qubit is arbitrary, the properties of the system must not change if these labels are assigned the other way round. Thus, the properties of each lattice configuration are identical to those of the

configuration in which all physical qubits have opposite spin. Consequently, the number of actual configurations reduces to $2^{2L_xL_y-1}$. Each quantum state, in which the lattice can be, is a superposition of these configurations. This is also true for the ground state, which is characterized by being the quantum state with the lowest energy that also is an eigenfunction of the Hamiltonian. It therefore has to be one of these configurations or has to consist of a superposition of them.
The best approach to find the ground state is to split up the Hamiltonian in two parts, the sum over sites and the sum over lattices:

$$H = H_{\text{Sites}} + H_{\text{Plaquettes}}$$
$$H_{\text{Sites}} = -\sum_{i \in \text{Sites}} J \cdot X_{+,i} \quad , \quad H_{\text{Plaquettes}} = -\sum_{j \in \text{Plaquettes}} J \cdot Z_{\square,j} \tag{2.2.9}$$

When only looking at H_{Sites}, $4 \cdot 2^{L_xL_y-1}$ configurations can be determined that yield the lowest energy, i.e. where the application of each $X_{+,i}$ operator returns 1. The simplest of these configurations $C_{T,0}$ is the one in which the spins of all the physical qubits are in the state $\sigma^1 = 1$. Thus

$$C_{T,0} = \bigotimes_{\ell \in \text{links}} |1_\ell\rangle . \tag{2.2.10}$$

The other configurations are similar to this one, just that they contain one or several closed chains of links in the state $\sigma^1 = -1$. These are sets of adjacent links which form a closed loop.
The configurations can be divided into four classes. The first class consists of $C_{T,0}$ as well as all configurations that can be created thereof by applying the different combinations of the $Z_{\square,j}$ operators, i.e. the configurations given by

$$C_T = \left(\prod_{j \in \substack{\text{some of the} \\ \text{plaquettes}}} Z_{\square,j} \right) C_{T,0} . \tag{2.2.11}$$

The second class forms around the configuration $C_{H,0}$, where at all links $\sigma^1 = 1$, except for links along a strictly horizontal closed chain on the lattice, where $\sigma^1 = -1$:

$$C_{H,0} = \left(\prod_{\ell \in \substack{\text{links along a} \\ \text{closed strictly} \\ \text{horizontal chain}}} \sigma_\ell^3 \right) C_{T,0} \tag{2.2.12}$$

This second class consists of $C_{H,0}$ as well as all configurations that can be created thereof by applying the different combinations of the $Z_{\square,j}$ operators. These can be expressed as

$$C_{H,i} = \left(\prod_{j \in \substack{\text{combination } i \\ \text{of the plaquettes}}} Z_{\square,j} \right) \left(\prod_{\ell \in \substack{\text{links along a} \\ \text{closed strictly} \\ \text{horizontal chain}}} \sigma_\ell^3 \right) C_{T,0} . \tag{2.2.13}$$

The third class forms around a similar configuration as $C_{H,0}$, namely $C_{V,0}$. The difference is that here the closed chain is strictly vertical:

$$C_{V,0} = \left(\prod_{\ell \in \substack{\text{links along a} \\ \text{closed strictly} \\ \text{vertical chain}}} \sigma_\ell^3 \right) C_{T,0} \tag{2.2.14}$$

This third class consists of $C_{V,0}$ as well as all configurations that can be created thereof by

applying the different combinations of the $Z_{\square,j}$ operators. These can be written as

$$C_{V,i}=\left(\prod_{j\in \substack{\text{combination } i\\ \text{of the plaquettes}}} Z_{\square,j}\right)\left(\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{vertical chain}}} \sigma_\ell^3\right)C_{T,0}. \tag{2.2.15}$$

Finally, the fourth class forms around the configuration $C_{B,0}$, which has two such closed chains, one strictly horizontal, the other strictly vertical:

$$C_{B,0}=\left(\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{horizontal chain}}} \sigma_\ell^3\right)\left(\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{vertical chain}}} \sigma_\ell^3\right)C_{T,0} \tag{2.2.16}$$

This fourth class consists of $C_{B,0}$ as well as all configurations that can be created thereof by applying the different combinations of the $Z_{\square,j}$ operators. These can be expressed as

$$C_{B,i}=\left(\prod_{j\in \substack{\text{combination } i\\ \text{of the plaquettes}}} Z_{\square,j}\right)\left(\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{horizontal chain}}} \sigma_\ell^3\right)\left(\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{vertical chain}}} \sigma_\ell^3\right)C_{T,0}. \tag{2.2.17}$$

The closed strictly horizontal respectively vertical chains that distinguish the accentuated configurations $C_{T,0}, C_{H,0}, C_{V,0}$ and $C_{B,0}$ are called *Polyakov loops* (see also Figure 2.2.3). The corresponding operators P_i that creates these loops apply σ^3 on the links of a strictly horizontal respectively vertical chain:

$$P_H=\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{horizontal chain}}} \sigma_\ell^3 \quad , \quad P_V=\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{vertical chain}}} \sigma_\ell^3 \tag{2.2.18}$$

Since they consists of σ^3 operators, the P_i commute with the Z-plaquette operators. Because at each site of the lattice they apply σ^3 on either none or two adjacent links, they also commute with the X-star operators. Hence, they commute with the Hamiltonian H as well. As it was shown before, these operators transform a configuration from one class into that of another. Two types of Polyakov loop operators can be distinguished, horizontal P_H respectively vertical P_V ones. The first adds a horizontal Polyakov loop transforming a first (trivial) class state into a second (horizontal) class one, respectively a third (vertical) class state into a fourth (both) class one. The second operator adds a vertical Polyakov loop and transforms a first (trivial) class state into a third (vertical) class one, respectively a second (horizontal) class state into a fourth (both) class one.

The Polyakov loop operators have dual analogues A_i (see also Figure 2.2.3). These operators apply the σ^1 operator on all the links that form the corresponding Polyakov loop on the dual lattice:

$$A_H=\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{horizontal chain}\\ \text{on the dual lattice}}} \sigma_\ell^1=\prod_{\ell\in \substack{\text{vertical links}\\ \text{along a row}\\ \text{of the lattice}}} \sigma_\ell^1 \quad , \quad A_V=\prod_{\ell\in \substack{\text{links along a}\\ \text{closed strictly}\\ \text{vertical chain}\\ \text{on the dual lattice}}} \sigma_\ell^1=\prod_{\ell\in \substack{\text{horizontal links}\\ \text{along a column}\\ \text{of the lattice}}} \sigma_\ell^1 \tag{2.2.19}$$

This means that the horizontal dual analogue operator A_H applies σ^1 to all vertical links contained inside a horizontal row of plaquettes, while the vertical analogue operator A_V applies

σ^1 to all horizontal links of a vertical column of plaquettes. With these operators the class to which a configuration belongs can be determined. Applying them on a configuration yields

$$\begin{aligned} A_H C_{T,i} &= +C_{T,i}, & A_V C_{T,i} &= +C_{T,i}, \\ A_H C_{H,i} &= +C_{H,i}, & A_V C_{H,i} &= -C_{H,i}, \\ A_H C_{V,i} &= -C_{V,i}, & A_V C_{V,i} &= +C_{V,i}, \\ A_H C_{B,i} &= -C_{B,i}, & A_V C_{B,i} &= -C_{B,i}. \end{aligned} \tag{2.2.20}$$

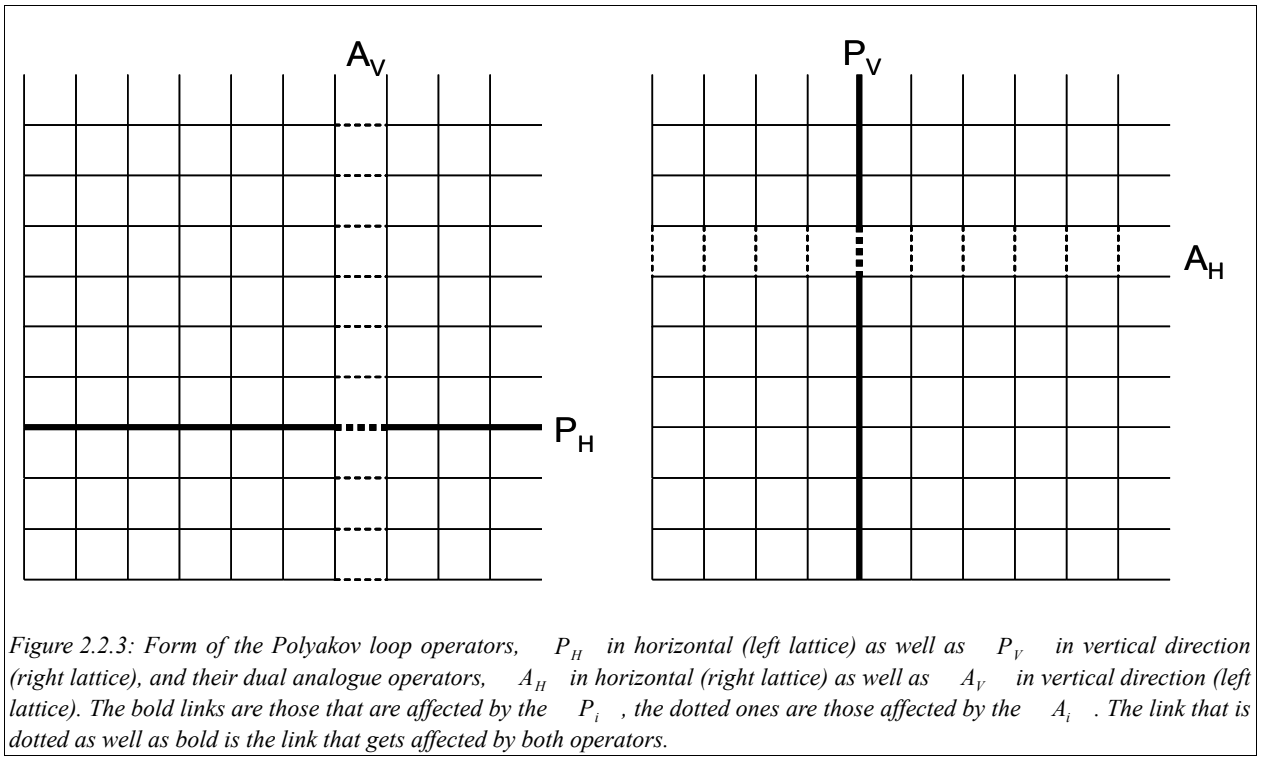

Figure 2.2.3: Form of the Polyakov loop operators, P_H in horizontal (left lattice) as well as P_V in vertical direction (right lattice), and their dual analogue operators, A_H in horizontal (right lattice) as well as A_V in vertical direction (left lattice). The bold links are those that are affected by the P_i, the dotted ones are those affected by the A_i. The link that is dotted as well as bold is the link that gets affected by both operators.

Since the A_i operators consist of applications of σ^1, they commute with the X-star operator. Since each Z-plaquette operator either overlaps at none or two links with the A_i operators, they commute as well. Thus, the A_i operators also commute with the Hamiltonian H. However, they do not commute with the Polyakov loop operators P_i, because these two operators overlap at only one link and there apply different Pauli matrices (dashed bold links in Figure 2.2.3).
While the aforementioned configurations and their linear combinations all are ground states of H_{Sites}, they are not necessarily ground states of H as well. For the latter, they also need to form a ground state of $H_{\text{Plaquettes}}$, which is equivalent to fulfilling the Gauss law of the $\mathbb{Z}_2$ gauge symmetry of the dual lattice. This means that the ground state $|\Psi_0\rangle$ of the toric code has to fulfil the following relation for every gauge transformation, i.e. for the application of every possible combination k of Z-plaquette operators $Z_{\square,j}$

$$\prod_{\substack{j \in \text{ plaquettes in} \\ \text{combination } k}} Z_{\square,j} |\Psi_0\rangle \stackrel{!}{=} |\Psi_0\rangle. \tag{2.2.21}$$

This constraint is only fulfilled for four combinations of the configurations as well as linear combinations of them. They are

$$\begin{aligned} |\Psi_{0,T}\rangle &= 2^{\frac{1-L_x L_y}{2}} \cdot \sum_{i \in \substack{\text{configurations} \\ \text{in the first} \\ \text{(trivial) class}}} C_{T,i} \quad , \quad |\Psi_{0,H}\rangle = 2^{\frac{1-L_x L_y}{2}} \cdot \sum_{i \in \substack{\text{configurations} \\ \text{in the second} \\ \text{(horizontal) class}}} C_{H,i}, \\ |\Psi_{0,V}\rangle &= 2^{\frac{1-L_x L_y}{2}} \cdot \sum_{i \in \substack{\text{configurations} \\ \text{in the third} \\ \text{(vertical) class}}} C_{V,i} \quad , \quad |\Psi_{0,B}\rangle = 2^{\frac{1-L_x L_y}{2}} \cdot \sum_{i \in \substack{\text{configurations} \\ \text{in the fourth} \\ \text{(both) class}}} C_{B,i}. \end{aligned} \tag{2.2.22}$$

These four combinations thus form the ground state of the toric code, which is fourfold degenerate. This allows two logical qubits to be encoded in one lattice.

2.2.3. Error theory of toric codes

The error theory of surface codes is a theoretical model that describes how errors occur in the calculation of a surface-code-based quantum computer and how these errors will influence the outcome.

It is assumed that external sources are able to cause two types of errors on the physical qubits of a surface code, *bit-flips* and *phase-shifts*. If the information storage is based on the σ^1 eigenstates of the physical qubits, as it was the case in the previous subsection, the bit-flip errors are equivalent to applications of a Pauli-Z-operator σ^3 on a specific physical qubit and the phase-shift errors are equivalent to the applications of a Pauli-X-operator σ^1.

Furthermore, it is assumed in this theory that the errors occur stochastically, that each error occurs independent of the other ones and that the different types of errors are uncorrelated. This allows the assignment of probabilities for the errors to occur, as well as a separate discussion of bit-flip and phase-shift errors. For simplicity, it is assumed that the different types of errors are equally likely to occur.

2.2.4. Error diagnosis and correction of toric codes

The problem of error diagnosis in quantum information storage devices is that such a diagnosis consists of a measurement of the quantum system. In general, the state of a quantum system is changed by a measurement. The only exception to this are measurements which correspond to operators of which the current state of the system is an eigenstate. Thus the error diagnosis, in order not to interfere with the stored data, must use exclusively such measurements. On the other hand, these measurements must also provide some sort of information on the correctness of the stored data.

For toric codes, where the system is in a state that consists of a superposition of the four ground states (see eq. (2.2.22)) with some bit-flip and phase-shift errors, there are two types of measurement operators that fulfil these requirements, the X-star and the Z-plaquette operator (see Section 2.2.2 and Figure 2.2.2), which therefore are also referred to as *diagnosis operators*. The X-star operator applies the Pauli-X operator σ^1 on the four links adjacent to a specific site on the lattice. It is able to detect an odd number of errors that are caused by the application of σ^3 on these links (bit-flip errors, if the information storage is based on the σ^1 eigenstates of the physical qubits). On the other hand, the Z-plaquette operator applies the Pauli-Z operator σ^3 on the four links forming a plaquette. It is able to detect an odd number of errors that are caused by the application of σ^1 on these links (phase-shift errors, if the information storage is based on the σ^1 eigenstates of the physical qubits). As it was shown in Section 2.2.2, the X-star and Z-plaquette operators all commute with one another. Furthermore, the ground state and the erroneous states are eigenstates of these operators. They thus can all be measured simultaneously

without disturbing the stored information or the measurement outcome of each other.
This property motivates the strategy to check for errors in the lattice. In regular time intervals, such a simultaneous measurement of all X-star and Z-plaquette operators is performed. The resulting information gathered from these measurements, the so called *error syndrome*, is stored in a classical data storage and a classical computer is used to analyse it. An example of how such an error syndrome from the X-star measurements about the bit-flip errors could look like is given in Figure 2.2.4. As it can be easily seen, the erroneous links cannot be determined directly. Instead only information about lattice sites is obtained that are adjacent to an odd number of erroneous links. An isolated erroneous link can thus be determined by the measurement result of the two adjacent sites (see Figure 2.2.4a). In the case of a series of erroneous links of which each is adjacent to the previous one, only the sites at the two ends of this so called *error chain* can be detected (see Figure 2.2.4b). Furthermore, there are structures of erroneous links that cannot be detected (see Figure 2.2.4c&d). These undetectable structures have in common that they do not have open ends, the series of erroneous links forms a closed loop. They can be divided into two categories. Structures from the first category (Figure 2.2.4c) encircle plaquettes, structures from the second category (Figure 2.2.4d) do not. Since each pair of opposite boundaries of the lattice are joined together in the toric code, the error chains are able to wind around the lattice. The two aforementioned categories of error chains without ends differ in regard to these windings. The structures of the first category have an even number of windings in both horizontal and vertical direction. They correspond to gauge transformations. The structures of the second category have an odd number of windings in at least one of the directions. They correspond to a gauge transformation combined with the application of a Polyakov loop operator in the direction(s) where the number of windings is odd.

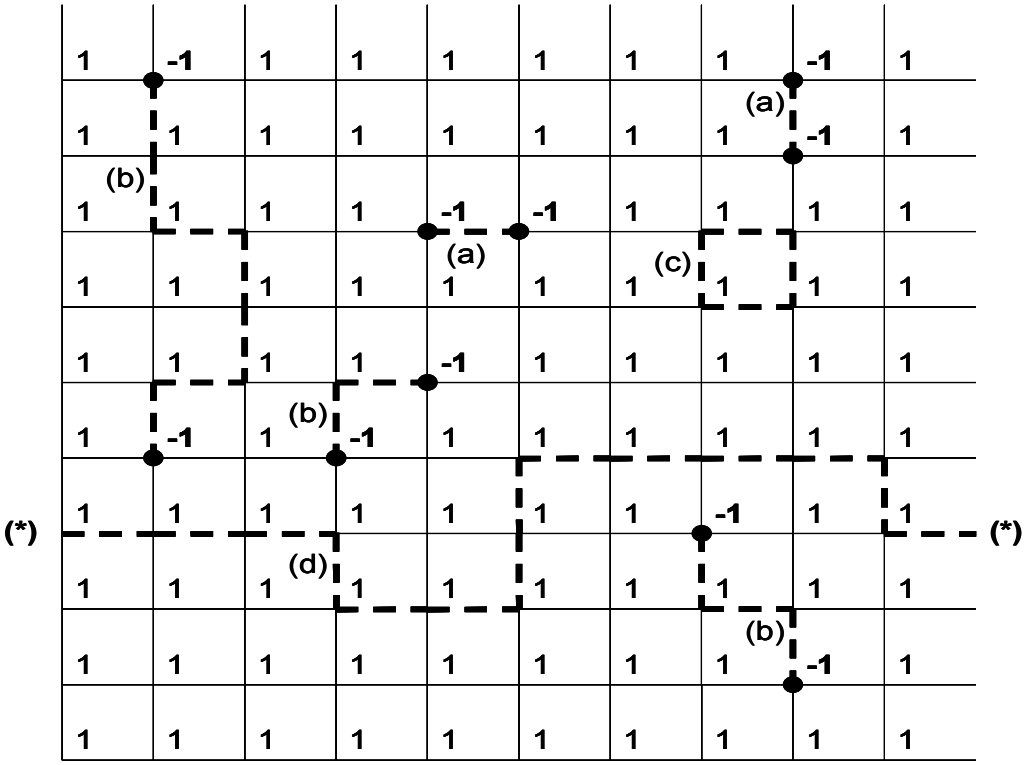

Figure 2.2.4: Example of how bit-flip errors could be located on a toric code lattice. Erroneous links are highlighted using a dashed line, non-erroneous links are drawn with a continuous line. The numbers next to each site of the lattice show the results of the error syndrome's X-star measurements. This measurement result is 1, if there are none or an even number of erroneous links adjacent to the site, and -1, if the number of adjacent erroneous links is odd. The later ones are furthermore marked with a black dot for easier recognition. Keep in mind that when actually using the error correction technique discussed here, only the results of the syndrome measurements are known. The position of the erroneous links, different than shown here, cannot be directly determined.
The erroneous links form different types of structures. These are labelled:
(a) isolated error: a single erroneous link without another erroneous link adjacent to it
(b) open error chain: a series of erroneous links of which each is adjacent to the next and which has a distinct beginning and end
(c) category 1 closed error chain: a series of erroneous links of which each is adjacent to the next one and which encircles one or several plaquettes. If it winds around the lattice in a direction, it does this an even number of times.
(d) category 2 closed error chain: a series of erroneous links of which each is adjacent to the next one and which does not encircle any plaquettes. It winds around at least one direction of the lattice an odd number of times. Notice that due to the geometry of the toric code the two positions of the error chain shown in the figure which are marked with (*) are actually connected.

Due to the gauge symmetry of the toric code all structures of the first category of error chains without ends leave the quantum state of the lattice invariant. This is because the application of these error chains on the ground state bijectively maps in each class of configurations every configuration onto another configuration of the same class. Since the basis states of the ground state of the lattice are equal-weight superpositions of all configurations of the corresponding class, the basis states get mapped onto themselves. The quantum state of the lattice and the information stored within thus remain unaffected.
The aforementioned effect can be used in the strategy to correct the errors in the toric code. After each round of syndrome measurement, which is assumed to be performed without error, the measurement data is inserted into a classical computer. There an algorithm is executed which finds the most probable positions of erroneous links on the lattice that leads to the measured error syndrome. Because the errors are assumed to occur stochastically with a given probability and because the measurement results of the error syndrome only reveal the end points of open error chains, the algorithm simulates the error formation in order to connect these end points into pairs of two using as few links as possible. To do so, links for the correction chain are selected with the same probability as errors occur. This makes a correction chain more unlikely to be used the longer it is. The links of the correction chain are assumed to be the erroneous links. In the next step, an operator is applied to these links in order to correct the errors. In the case of bit-flip errors, this is the σ^3 operator, since it is its own inverse. There are three possible outcomes to this correction step (see Figure 2.2.5):
In the first case, the algorithm correctly determines a complete error chain (Figure 2.2.5A). Hence, the application of the σ^3 operator reverses the errors along this chain restoring the non-erroneous state of the links.
In the second case, the algorithm connects the two end points of an open error chain. However, it does not determine the correct position of all contained erroneous links. Instead the computed error chain connects the ends by a different path (Figure 2.2.5B). This case is quite likely, because in most cases there are several paths of the same length that connect two sites on the lattice. Thus, there is no criterion that allows the distinction of the actual path of the error chain from one that has the same length but goes a different way. The result of the correction step belonging to this case is that the error chain gets extended by the application of the σ^3 operator to previously non-erroneous links. An exception are the links which have been correctly determined erroneous and thus get corrected by the σ^3 operator. In addition to that, the previously open ended error chain has transformed into a closed error chain. If this closed error chain falls into the first category, due to the gauge symmetry the non-erroneous ground state of the lattice is restored. However, if the closed error chain falls into the second category, the Polyakov loop it forms will change the state of the quantum system. This means that the data stored in the lattice has been altered without leaving any measurable signs indicating that such an alteration has occurred and needs to be corrected. This means the stored data got irreversibly damaged.
In the third case, the algorithm connects end points of distinct open error chains (Figure 2.2.5C). This leads to an extension of the involved error chains as well as their combination into one closed error chain. Such a closed error chain takes on one of the two forms presented in the second case. The consequences of this are the same as discussed above.
To summarize, it can be said that this method of correction is successful as long as the error chain by itself or together with the correction chain does not form a closed error chain of the second category. If it does, the stored data has undetectably been altered and thus irreversibly damaged. The calculation process, which uses this data, thus is not performed correctly and yields a wrong result. If this occurs too frequently, a toric code is not sufficiently reliable to be used as a quantum data storage device. There are, however, two strategies to decrease the probability of this type of closed error chain to form. The first strategy is to increase the lattice size. The larger the lattice is, the more links have to become erroneous in order to form an error chain that makes a complete

winding around the lattice. Since each link only becomes erroneous with a certain probability, the likelihood for the formation of an error chain decreases with increasing length of that chain. The second strategy is to decrease the probability with which a link becomes erroneous. As this probability is defined as the likelihood that a link changes its state in the time between two consecutive error syndrome measurements, this can, for example, be achieved by measuring the error syndrome more often. This decreases the amount of time in which the errors can occur. Since the errors are caused by external effects independent of the error syndrome measurements, it can be assumed that their temporal probability distribution to occur is uniform in time. Thus, during a shorter time span between two error syndrome measurements fewer errors will manifest themselves.

So far, only the correction of the bit-flip errors using the data from X-star operator measurements has been discussed. The correction of the phase-shift errors using the data from the Z-plaquette measurements, however, turns out to work analogously on the dual lattice. The plaquettes of the lattice form the sites of the dual lattice. Furthermore, the duality transformation rotates all links by 90°. Thus, the Z-plaquette operators on the lattice turn into Z-star operators on the dual lattice. With the right coordinate transformation on the qubit quantum state, σ^1 can be turned into σ^3 and vice versa. Hereby the phase-shift error correction problem is translated into the bit-flip error correction problem discussed above.

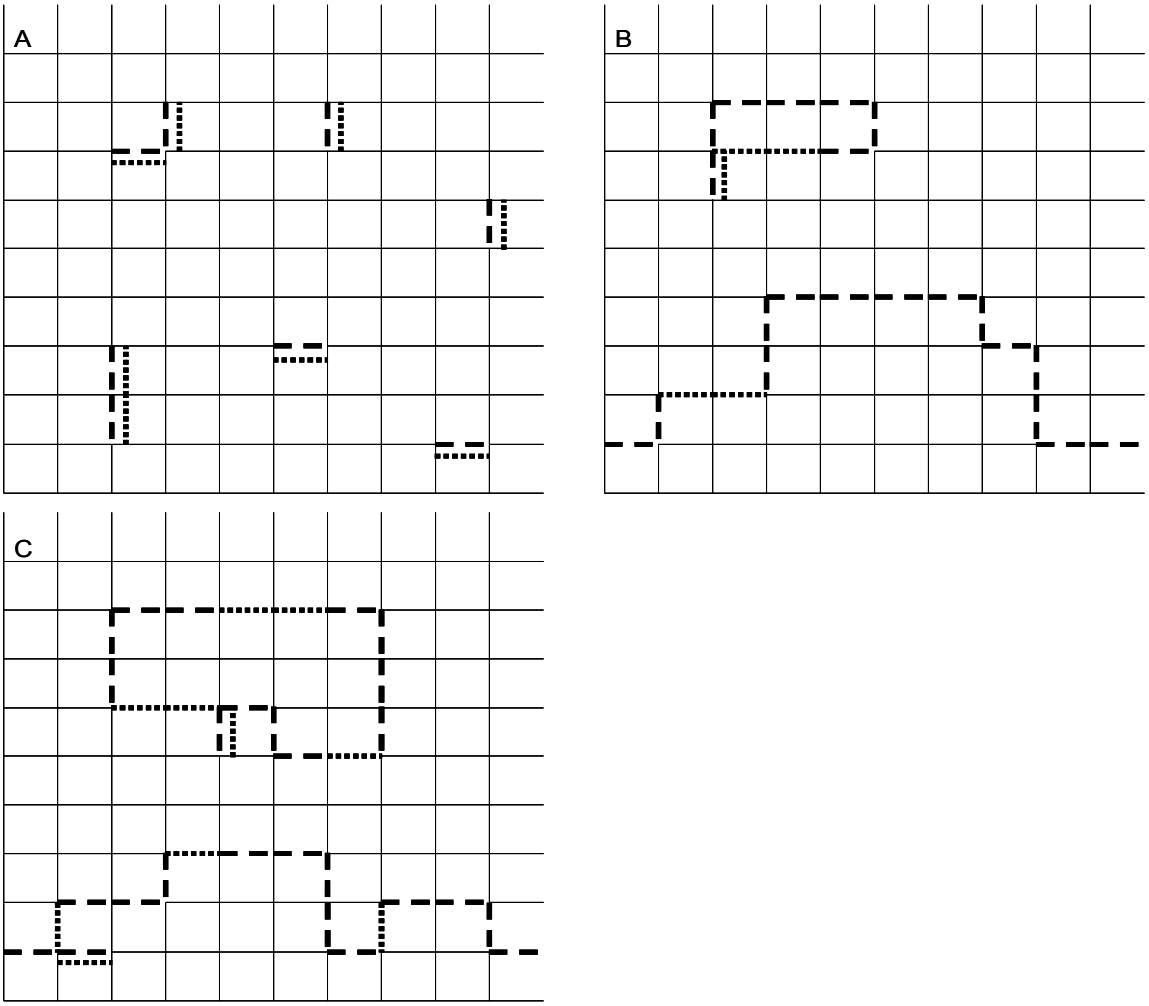

Figure 2.2.5: Different toric code lattices with open error chains (dashed lines) and chains of links on which the corresponding Pauli operator is applied in the effort to correct them (dotted lines). Note that on the links where both chains overlap, the dotted line is not drawn on the link, but next to it for reasons of clarity.
On lattice A, all error chains have been determined correctly and in the correction step the corresponding Pauli operator is applied on all the erroneous links. Thus all errors are removed.
On lattice B, the correction step has connected the end points of the distinct open error chains. However, the correcting chains do not follow the error chains. The result is two closed error chains. The top one belongs to the first category. Thus there the errors get corrected. The bottom one, however, belongs to the second category. It therefore undetectably modifies the stored information, damaging it irreversibly.
On lattice C, in the correction step the end points of different open error chains have been connected. This has lead to the formation of two closed error chains. The top one again belongs to the first, the bottom one to the second category. Hence in the upper chain the errors get corrected while the lower chain undetectably modifies the stored information and damages it irreversibly.

2.2.5. Relation to the random-bond Ising model

As outlined in E. Dennis et al. [2], the error correction of the surface codes is related to the two-dimensional *random-bond Ising model* (RBIM). This relation can be used to translate findings from this thermodynamic model to questions regarding errors in the toric code. But in order to do so, first the RBIM needs to be described and the relation discussed:
The Ising model is a classical model that thermodynamically describes a microscopic system which forms a magnet. It consists of an array of particles each of which has a classical spin. This spin can be either +1 or -1. The system can be in different states depending on the orientation of the individual spins. The energy of a certain configuration is given by the classical Hamilton function $\tilde{H}$, which is written as

$$\tilde{H} = -\sum_{<i,j>} J_{ij} \cdot s_i \cdot s_j . \tag{2.2.23}$$

Where $s_i = \pm 1$ are the spins of the individual particles, $J_{ij} \in \mathbb{R}$ is the coupling between two particles and the sum goes over all pairs of neighbouring particles (to be called *bonds*).
As long as the couplings between two neighbouring spins J_{ij} are mostly positive, the system energetically prefers to stay in a state where most spins have the same orientation. However, when more couplings J_{ij} are negative, the system energetically prefers to be in a state where the orientation of the spins alternate. A positive J_{ij} between two spins is also called a *ferromagnetic bond*, while a negative J_{ij} is also referred to as an *antiferromagnetic bond*. A special case of the Ising model is the RBIM. Here the following constraint is put on the coupling:

$$J_{ij} \in \{-J, +J\} \tag{2.2.24}$$

where J is a positive constant. Consequently, this model has only two types of couplings, a ferromagnetic and an antiferromagnetic type. Both have the same strength. To each bond its type is assigned randomly.
The RBIM has two free parameters. The first is temperature, also referred to as the inverse of the inverse temperature β^{-1}. The second is the probability to have an antiferromagnetic bond p_{AFB}. Variations in these parameters result in the behaviour that is depicted in the phase diagram shown in Figure 2.2.6. As it can be easily seen, there are two different phases, a ferromagnetic and a paramagnetic one. The ferromagnetic phase forms at sufficiently low temperatures and probabilities p_{AFB}. Here most spins are aligned, the system energetically prefers an ordered state. In contrast, the paramagnetic phase forms once at least one of the two free parameters becomes too big. Here the system energetically prefers to be in a disordered state with hardly any correlation between the orientations of the spins.
The analogy to the toric code error correction arises from treating the plaquettes like the spin particles of the two-dimensional RBIM, non-erroneous links like ferromagnetic bonds, erroneous links like antiferromagnetic bonds and Polyakov loops like antiperiodic boundary conditions. Due to this, the probability p_{AFB} to have an antiferromagnetic bond maps on the probability p that an error forms on a link. Furthermore, the correction chains of the toric code error correction model correspond to different spin configurations with distinct periodic/antiperiodic boundary conditions in the RBIM. Two anti-aligned spins over an antiferromagnetic bond correspond to an erroneous link that has been correctly recognized and corrected, two anti-aligned spins over a ferromagnetic bond correspond to a link that is just part of the correction chain, even though it is not erroneous.

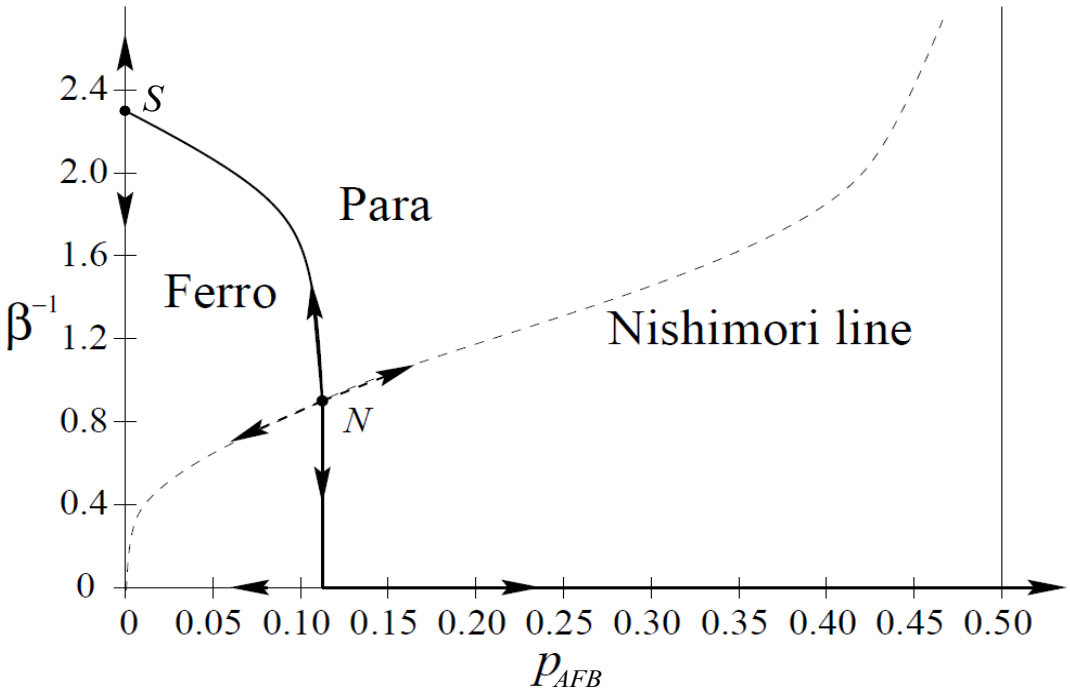

Figure 2.2.6: Phase diagram of the random-bond Ising model. The horizontal axis shows the probability p_{AFB} that the bond between two neighbouring spins is antiferromagnetic instead of ferromagnetic. The vertical axis shows the inverse of the inverse temperature β^{-1}, i.e. the temperature. The continuous line indicates the phase boundary between the ferromagnetic and the paramagnetic phase. Furthermore, the Nishimori line (dashed line) and the Nishimori point N (intersection between phase boundary and Nishimori line) are marked. The behaviour along the β^{-1}-axis is that of the standard Ising model. Its critical point, where it crosses the phase boundary, is marked with S. [2]

However, in the analogy the inverse temperature β is not a free parameter. Instead, it depends on the error formation probability p as well. The relation between them is given by

$$e^{-2\beta} = \frac{p}{1-p} \tag{2.2.25}$$

In the phase diagram, this relation forms the so called *Nishimori line*. This line crosses the phase boundary at the point referred to as the *Nishimori point* N.
The analogue of the ferromagnetic phase is a toric code lattice where the majority of links is non-erroneous. In the case of a lattice of infinite size, this would lead to all closed error chains belonging to the first category. The reason for this is that for a closed error chain of the second category, it would have to wind around the lattice in at least one direction. To do so, however, an infinite number of erroneous links is required, far too many if the majority of links has to be non-erroneous. Thus, in the analogue of the ferromagnetic phase, the errors in a toric code with infinite lattice size will always be eliminated correctly. On the other hand, in the analogue of the paramagnetic phase, such second category closed error chains become possible on infinite size lattices. Thus here, the correction of the errors will fail with a non-vanishing probability.
Uniting the findings of the previous paragraphs yields the following conclusion: The Nishimori point lies at a critical probability for error formation p_{crit} which marks the phase transition in the toric code between a phase where on a lattice of infinite size all errors are always corrected, and another phase where this is no longer the case. This critical probability is also known as the *accuracy threshold*. Hence, if the toric code error correction is designed in a way in which the probability for error formation between the different rounds of error determination lies below this critical value, the erroneous links will be corrected with a good reliability, when the lattice size is chosen sufficiently large.

3. Method

The objective of this thesis is to determine the accuracy threshold p_{crit}, the critical probability of error formation, where a toric code lattice of infinite size undergoes the phase transition from a phase where the errors are always eliminated correctly to a phase where this is no longer the case (see Section 2.2.5). This was done numerically using two different versions of a worm algorithm. First these algorithms were tested on the standard Ising model (SIM). Thereafter, they were extended to a simulation of the random-bond Ising model (RBIM).
This section first presents in Subsection 3.1 the analytical preparations that were necessary to be able to translate the problem from toric code error correction to the RBIM. Thereafter, in Subsection 3.2 it describes in detail the two types of the worm algorithm used to perform the numerical calculation as well as the algorithms in which these worm algorithms were embedded to simulate the SIM respectively RBIM. Finally, it lists the different series of calculations that were made, together with their parameters, in Subsection 3.3.

3.1. Analytical preparation

The error diagnosis and correction method on the toric code, which is described in detail in Section 2.2.4, works by first determining the ends of the open error chains. Then, in the next step, a set of links is chosen that combines these ends with one another. By applying the operator that caused the error on these chosen links, the open error chains are extended and closed. Depending on which category (see Section 2.2.4 for the definition) the closed error chain falls into, i.e. whether or not a Polyakov loop operator is applied, this either corrects the information stored on the lattice or irreversibly damages it. Since the lattice is a two-dimensional structure (it has a horizontal and a vertical direction) the different resulting configurations of the lattice can be divided into four homology classes:

- trivial: All closed error chains are of the first category or there is an even number of second category closed error chains in both horizontal and vertical direction.
- horizontal: There is an odd number of second category closed error chains in the horizontal direction. If they are removed, the remaining closed error chains form a configuration that belongs to the trivial homology class.
- vertical: There is an odd number of second category closed error chains in the vertical direction. If they are removed, the remaining closed error chains form a configuration that belongs to the trivial homology class.
- both: There is an odd number of second category closed error chains in the horizontal as well as in the vertical direction. All other closed error chains are of the first category.

These homology classes look similar to the four classes of configurations in Section 2.2.2 that form the distinct basis states of the ground state. However, they are not the same and need to be distinguished. In Section 2.2.2, the configurations of each class get together in a equal-weight superposition to form the four basis states of the ground state. On the other hand, here the homology classes represent different types of operators that act differently upon these basis states. They do not form superpositions but instead can act separately on the ground state of the lattice. Those of the trivial homology class correspond to the gauge transformations. Thus, they do not change the ground state. The other homology classes correspond to gauge transformations combined with applications of Polyakov loop operators. They affect the ground state.
Thus, only if the closed chain resulting from the correction attempt belongs to the trivial homology class, the information in the lattice is restored. If it belongs to the three other homology classes, it gets irreversibly damaged. Hence, the important question for information storage on a toric code lattice is: How likely will the formation of errors and their correction form a closed chain that belongs to the trivial homology class?

The answer to this question $P_{trivial}$ is given by the sum of the probabilities $\tilde{p}_i$ that a particular closed chain manifests, summed over all closed chains in the trivial homology class:

$$P_{trivial} = \sum_{i \in \text{trivial homology class}} \tilde{p}_i \tag{3.1.1}$$

This sum can be split up into a product of two sums, one over all different combinations of errors that can manifest themselves, the other over all ways the formed open error chains can be closed to form a closed error chain of the first category. Furthermore, the probability can be split into the product of a part that depends on the error combination $\hat{p}_i$ and another part that depends on the applied correction $\hat{p}_j$:

$$P_{trivial} = \sum_{i \in \begin{array}{l}\text{combinations of erroneous}\\ \text{links on the lattice}\end{array}} \; \sum_{j \in \begin{array}{l}\text{ways to combine}\\ \text{the open ends in } i\\ \text{so that the resulting}\\ \text{final state belongs}\\ \text{to the trivial}\\ \text{homology class}\end{array}} \hat{p}_i \cdot \hat{p}_j \tag{3.1.2}$$

As discussed in Section 2.2.5, the probability p to have an erroneous link in the toric code is analogous to the probability p_{AFB} of having an antiferromagnetic bond in the RBIM:

$$p \overset{!}{=} p_{AFB} \tag{3.1.3}$$

Consequently, the probability $\hat{p}_i$ to have a certain configuration i of erroneous links in the lattice of the toric code corresponds to the probability $p_{AFB,i}$ of having the configuration i of antiferromagnetic bonds in the RBIM. Thus

$$\begin{aligned}\hat{p}_i &= p^{\text{number of erroneous links in } i} \cdot (1-p)^{\text{number of non-erroneous links in } i} \\ &\overset{!}{=} p_{AFB}^{\text{number of antiferromagnetic bonds in } i} \cdot \left(1-p_{AFB}\right)^{\text{number of ferromagnetic bonds in } i} = p_{AFB,i}\end{aligned} \tag{3.1.4}$$

Furthermore, the probability $\hat{p}_j$ to have a certain correction chain applied on the toric code corresponds to the probability to realize a specific spin configuration C_j on the lattice of the RBIM:

$$\hat{p}_j = p_{C_j} \tag{3.1.5}$$

In doing so, the four different homology classes of the toric code get translated into the four different combinations of periodic respectively antiperiodic boundary conditions along the two directions of the lattice in the RBIM. The trivial homology class of the toric code matches the RBIM with periodic boundary conditions in both directions. The other homology classes of the toric code match the RBIM with antiperiodic boundary conditions in horizontal, vertical respectively both directions. A system configuration in the RBIM with given p_{AFB} and β^{-1} belongs to the ferromagnetic phase, if for it the proportion of the weights of configurations with periodic boundary conditions is predominant over the sum of its three other proportions representing the weights of configurations with antiperiodic boundary conditions. If this is no longer the case, the phase is paramagnetic. Thus, the phase of the toric code error correction, where the errors get corrected successfully using the method described in Section 2.2.4, corresponds to the ferromagnetic phase of the RBIM while the phase, where the error correction method breaks down, corresponds to the paramagnetic phase of the RBIM.

In consequence, $P_{trivial}$ can be expressed as

$$P_{trivial} = \sum_{i \in \text{combinations of links on the lattice}} p_{AFB,i} \sum_{j \in \text{combinations of plaquettes on the lattice with periodic boundary conditions}} p_{C_j,i} \tag{3.1.6}$$

Note that the probability $p_{C_j,i}$ to realize a specific spin configuration also depends on the configuration of antiferromagnetic bonds that is present on the lattice.
For each of these four types of boundary conditions in the RBIM the same set of configurations exists. The configurations in each set differ by the position and/or number of antiferromagnetic bonds as well as their spin configuration. The probability with which each configuration is realized thus depends on the probability p_{AFB} with which an antiferromagnetic bond forms and the temperature β^{-1} of the system as well as the boundary conditions. Due to the nature of the used algorithms, it is useful to consider the spin change probability p_{SC} of the system instead of the temperature. In this model, there are different realizable configurations with different energies. The algorithms perform transitions from one of these configurations C_i to another one C_j. Each of these transitions occurs with a certain probability. This is given by

$$P_{transition}(C_i \to C_j) = \begin{cases} 1, & E(C_j) < E(C_i) \\ p_{SC}, & E(C_j) > E(C_i) \end{cases}. \tag{3.1.7}$$

Where $E(C_i)$ is the energy of the configuration C_i. The transition thus is always performed, if it leads to an energetically lower configuration. If it causes a rise in the system's energy, the transition will only happen with probability p_{SC}. Since the energy of the system is quantized so that the energy difference $|E(C_i) - E(C_j)|$ between the initial and final configuration of the transition is always the same, this probability is related to the temperature by

$$p_{SC} = e^{-2\beta} \tag{3.1.8}$$

as it can be easily looked up in a statistical thermodynamics textbook [3].
As it can be deduced from eq. (2.2.23), the energy of a certain configuration $E(C_i)$ is given by the orientation of neighbouring spins and the type of the associated bonds. Aligned neighbouring spins over a ferromagnetic bond and anti-aligned neighbouring spins over an antiferromagnetic bond decrease the energy of the system, while anti-aligned neighbouring spins over a ferromagnetic bond and aligned neighbouring spins over an antiferromagnetic bond increase it. Furthermore, periodic boundary conditions behave like ferromagnetic bonds and antiperiodic boundary conditions like antiferromagnetic bonds in this respect. This allows putting the different configurations in an order with respect to their energies.
In addition, the described method allows the determination of the probability for a configuration to occur relative to the probability of a reference ground configuration. These relative probabilities can be understood as weights $W(C_i)$ of their configurations. If the reference ground configuration is chosen to be the configuration without any antiferromagnetic bonds where all spins are aligned in the same orientation, then the weights are given by

$$W(C_i) = \prod_{<k,m>} \tilde{p}_{SC}^{\frac{1}{2}\cdot\left(1 - sign(S_k)\cdot sign(S_m)\cdot sign(J_{km})\right)}. \tag{3.1.9}$$

Where the product goes over all pairs of neighbouring spins, $\tilde{p}_{SC}$ is the weight factor with which spin orientations contribute to the weight, $S_k = \pm 1$ are the individual spins, and J_{km} is the coupling between the spins S_k and S_m (see also eq. (2.2.23)) which also takes into account the boundary conditions. $\tilde{p}_{SC}$ is not necessarily identical to the spin change probability p_{SC}

mentioned above.
The relative probability $W_{AFB}(C_i)$, with which the antiferromagnetic bond pattern of a certain configuration C_i is realized from the reference ground configuration, is given by

$$W_{AFB}(C_i) = \prod_{j \in \text{links on the lattice}} \left(\frac{p_{AFB}}{1-p_{AFB}} \right)^{\delta_{\text{bond type of } j, \text{ antiferromagnetic}}} \tag{3.1.10}$$

where $\delta_{i,j}$ is the Kronecker delta.
With the weights of the individual configurations $W(C_i)$, the probability $P_{trivial}$ can be expressed as

$$P_{trivial} = \frac{\sum_{i \in \text{configurations with periodic boundary conditions}} W(C_i)}{\sum_{j \in \text{configurations with all types of boundary conditions}} W(C_j)}. \tag{3.1.11}$$

Where the sum in the nominator goes over all the possible combinations of spin configurations with configurations of ferromagnetic and antiferromagnetic bonds for periodic boundary conditions. The sum in the denominator goes over all these combinations for the four different types of boundary conditions.
When, based on the analogy discussed in Section 2.2.5, the RBIM is used to describe the toric code error correction, the probability p_{AFB} to have an antiferromagnetic bond and the temperature β^{-1} (and thus also p_{SC}) are not independent. They have to fulfil the relation presented in eq. (2.2.25). In terms of p_{SC} this relation can be written as

$$p_{SC} = \frac{p_{AFB}}{1-p_{AFB}}. \tag{3.1.12}$$

In this analogy, the anti-alignment of spin orientations takes the role of the correction chain. Two anti-aligned spins over an antiferromagnetic bond correspond to an erroneous link that has been correctly recognized and corrected, two anti-aligned spins over a ferromagnetic bond correspond to a link that is just part of the correction chain, even though it is not erroneous. Since for the toric code error correction the links are assigned to the correction chain with a certain probability that equals the probability that these links are part of the error chain (see Section 2.2.4), the weight factor $\tilde{p}_{SC}$ of the spin orientations must also equal the weight factor for the formation of an antiferromagnetic bond (see eq. (3.1.10)). Hence

$$\tilde{p}_{SC} \stackrel{!}{=} \frac{p_{AFB}}{1-p_{AFB}}. \tag{3.1.13}$$

This allows writing eq. (3.1.11) as

$$P_{trivial} = \frac{\sum_{i \in \text{configurations with periodic boundary conditions}} \prod_{<k,m>} \left(\frac{p_{AFB}}{1-p_{AFB}} \right)^{\frac{1}{2} \cdot \left(1 - sign(S_k(C_i)) \cdot sign(S_m(C_i)) \cdot sign(J_{km}(C_i))\right)}}{\sum_{j \in \text{configurations with all types of boundary conditions}} \prod_{<k,m>} \left(\frac{p_{AFB}}{1-p_{AFB}} \right)^{\frac{1}{2} \cdot \left(1 - sign(S_k(C_j)) \cdot sign(S_m(C_j)) \cdot sign(J_{km}(C_j)\right)}} \tag{3.1.14}$$

To summarize, this means that the probability to have an antiferromagnetic bond p_{AFB}, which is equivalent to the probability of error creation in the toric code, is the only free parameter in this analogy in the RBIM. Based on it, on the one hand the likelihood of the formation of different ferromagnetic-antiferromagnetic bond patterns can be determined, on the other hand it sets the probability with which spins change to an energetically higher configuration. Furthermore, in the

here used analogy it determines the weights of the different configurations. These can be used to determine the probability $P_{trivial}$ with which a (based on the configuration weights) randomly chosen configuration belongs to the trivial homology class (in the toric code error correction picture) respectively with which it has only periodic boundary conditions (in the RBIM picture).
In order to check the part of the algorithm that creates the different spin configurations the standard Ising model (SIM) is used. While for the analogy the RBIM is reduced to the Nishimori line, the path in the p_{AFB} - β^{-1} -phase diagram given by eq. (2.2.25), the SIM is its reduction onto another path, namely one which follows the constraint

$$p_{AFB}=0\,. \tag{3.1.15}$$

This means that in the SIM there are only ferromagnetic but no antiferromagnetic bonds.
In the phase diagram of the RBIM (see Figure 2.2.6) this corresponds to a path following the β^{-1} -axis. Consequently the expression for $P_{trivial}$ simplifies to

$$P_{trivial}=\sum_{j\ \in\ \begin{array}{l}\text{ways to combine}\\ \text{the open ends in } i\\ \text{so that the resulting}\\ \text{final state belongs}\\ \text{to the trivial}\\ \text{homology class}\end{array}} \hat{p}_j \tag{3.1.16}$$

Analogously to above, this expression can be transformed into

$$P_{trivial}=\frac{\sum_{i\in\text{configurations with periodic boundary conditions}}\ \prod_{<k\,,\,m>}\left(\tilde{p}_{SC}\right)^{\frac{1}{2}\cdot\left(1-sign\left(S_k(C_i)\right)\cdot sign\left(S_m(C_i)\right)\right)}}{\sum_{j\in\text{configurations with all types of boundary conditions}}\ \prod_{<k\,,\,m>}\left(\tilde{p}_{SC}\right)^{\frac{1}{2}\cdot\left(1-sign\left(S_k(C_j)\right)\cdot sign\left(S_m(C_j)\right)\right)}}\,. \tag{3.1.17}$$

Here $\tilde{p}_{SC}$ is the only free parameter. For reasons of clarity, to better distinguish calculations in the two models, this parameter will be called q in the SIM.
For the two-dimensional SIM the location of the critical point separating the ferromagnetic from the paramagnetic phase can be analytically derived and is thus known exactly. This allows testing whether the part of the algorithm that simulates the different spin configurations of the Ising model works correctly. Based on Onsager [4], it can be derived that the phase transition in the two-dimensional SIM occurs at the critical inverse temperature of $\beta_{crit}=\frac{1}{2}\cdot\ln\left(\sqrt{2}+1\right)$ for lattices of infinite size. Using eq. (3.1.8), this value can be translated into the critical spin change probability

$$q_{crit}=p_{SC\,,crit}=\sqrt{2}-1\approx 0.41421\ldots\quad . \tag{3.1.18}$$

3.2. Algorithms

Since it is too expensive in labour and thus time to go through all possible configurations on a large lattice, and even impossible for a lattice of infinite size, another method has to be used that can yield the correct weighted distribution ratio of closed chains to the four homology classes within the bars of a certain error in an acceptable time frame as a numerical approximation.
One such method is the use of a so called *worm algorithm*. It starts at one configuration of the lattice and modifies it to another one. This is performed a large number of times, starting in a randomly created configuration of the trivial homology class. Since this algorithm is ergodic and fulfils detailed balance, as it is proven in Subsection 3.2.2, the probability to enter another specific configuration is proportional to the weight of this configuration. Therefore, configurations belonging to the different homology classes will be produced in a number proportional to the sum

of the weights of their configurations. Thus, after each application of the worm algorithm, the homology class, to which the produced configuration belongs, is determined. Then, it is counted how often configurations of the different homology classes have been created. The ratio between these counts gives the ratio between the sums of weights associated with each homology class. This enables the determination of the probability $P_{trivial}$ for a given probability p_{AFB} to have an antiferromagnetic bond using eq. (3.1.17).

The algorithm of the simulation of the RBIM first randomly generates patterns of antiferromagnetic bonds in the lattice. Thereafter, an initial configuration is set on the spins, followed by a variation of this spin configuration, which is able to change the boundary conditions as well. This variation is repeated several times and for each of the produced configurations, the homology class is determined. Thereafter this whole procedure is repeated for a new random pattern of antiferromagnetic bonds.

Since an antiferromagnetic bond between aligned spins behaves in the same way as a ferromagnetic bond between two anti-aligned spins or between two aligned spins separated by a boundary with anti-parallel boundary conditions, these cases do not need to be distinguished. Instead it is sufficient to call a link that raises the energy of the system *occupied*, while the other ones remain *unoccupied*. Thus, the initial antiferromagnetic/ferromagnetic bond pattern in the lattice, where links became antiferromagnetic with probability p_{AFB}, is realized using occupied/unoccupied links. The random initial spin configuration is achieved by randomly selecting plaquettes in that lattice with probability $p(iPS)$ and changing the link state of the links forming these plaquettes from occupied to unoccupied respectively vice versa. Thereafter, the change from one spin configuration to another is achieved using the worm algorithm with $q=\frac{p_{AFB}}{1-p_{AFB}}$ (see eq. (3.1.13)) as the probability to change the state of a link from unoccupied to occupied (while the probability to change the link state from occupied to unoccupied is 1). In this process, the boundary conditions of the model will change every time the worm winds around the lattice in one direction, changing the homology class to which the next configuration belongs in the process.

The algorithm for the simulation of the SIM proceeds in almost the same way. The difference, however, is that here no randomly generated patterns of antiferromagnetic bonds are produced. Instead, only the spin configuration and the boundary conditions are changed. Thus, q can be used here directly as the input parameter.

In the next step, in both simulations the results of the homology class measurements are binned several times. In each binning the bins have a specific size. In the first binning there is 10^0=1 measurement per bin. The next binnings always have ten times more measurements per bin than the one before up until in the final binning all measurements are contained in a single bin. Then the counting is performed and the ratio between the different homology classes is determined inside each of these bins for all the binnings. By splitting up the measurement data in this way, many measurement values for the counts respectively the ratio values are obtained. This allows the determination of their standard deviation to be used as the error of these quantities. However, if the bins are too small, there is an autocorrelation between the contained measurements. This tends to produce an error estimate that is too small. On the other hand, if the bins are too large, there are only a few of them. This increases the size of the error estimate. Between these two extremes is an interval in bin size in which a good value for the error can be obtained. In order to find this intermediate region, it is determined for which of these binnings with different bin size that are adjacent the error remains more or less constant. The individual errors in this region are then averaged to form the error for the corresponding quantity.

In order to achieve a certain robustness in the produced results, the calculations are performed with two different types of worm algorithms and the results are then compared. Their differences are elaborated in Subsection 3.2.1. They are named *wormhead only* (WHO) respectively *wormhead- and tail* (WHAT).

3.2.1. Procedure of the worm algorithms

The worm algorithm starts by planting a so called *worm* on one random site in the lattice. This worm consists of two ends, the *wormhead* and *wormtail*, which are located on sites. They are connected by the *wormbody* consisting of a set of adjacent links on the lattice that start at the wormhead and finally reach the wormtail. Since the worm is initially planted at one site, its wormbody has a length of zero at that initial step. A non-zero length is acquired during the next steps of the algorithm. To do so, one of the four links adjacent to the initiation site, also referred to as *directions*, is selected at random. The probability to be chosen is the same for all four directions. Then, the state of the chosen link is determined. If it is occupied, the wormhead is moved along this link to the site the link connects to. Furthermore, the state of the link is switched to unoccupied. If the link, however, is unoccupied, the wormhead only moves along it with probability q. In this case the link state switches to occupied. With a probability $1-q$ the wormhead remains at the initiation site. In this case no link state is switched and one iteration of the algorithm is completed.
If the wormhead has been moved in the first step, the algorithm continues. Here the two versions of the algorithm, WHO and WHAT, start to differ. In the WHO-version of the algorithm, only the wormhead is moved in the following steps. In these, the previously described step is repeated on the new position of the wormhead. A direction is chosen at random. If this link is occupied, the wormhead moves along it switching the link state to unoccupied. If the link is unoccupied, the wormhead moves along it with probability q switching the link state to occupied or stays where it was with probability $1-q$ doing nothing and waiting for the next step. This is repeated until the wormhead meets the wormtail again. At this point one iteration of the algorithm is completed.
The WHAT-version of the algorithm works basically in the same way. The only difference is that it is not just the wormhead that can be moved, but both ends. Therefore, at the beginning of each step, one of these two ends is randomly selected with equal probability. It is then around this end, that a direction is chosen and it depends on the link state of these directions whether the chosen end is moved into one of these directions. Here as well, one iteration is completed once the two ends meet.
When the worm algorithm is applied several times, there is one more difference between the two versions. Since it would violate detailed balance to randomly move around the initiation site each time the algorithm is newly applied, the occurrence of this is only allowed with a probability strictly smaller than 1. Therefore, the WHO-version of the algorithm has been set in a way that from the second application onwards the initiation site is randomly selected only with a probability of 0.5. With a probability of 0.5, it remains the same as in the previous application. By this, detailed balance is still fulfilled, on the one hand, and on the other, the changes in the configuration produced by the worm algorithm do not restrict themselves to just a certain segment of the lattice. If a new initiation site gets selected randomly, the iteration is completed afterwards. It is not but at the next iteration that the worm gets a chance (with a probability 0.5) to propagate from this position. Since in the WHAT-version of the algorithm, both ends of the worm are able to move, here the initiation and termination site will likely be different from one another. Thus, the termination site of the previous application can be used as initiation site for the next application. This both satisfies detailed balance as well as prevents the changes in the configuration to be just in one segment of the lattice.

3.2.2. Proof of ergodicity and detailed balance

The proof of ergodicity, the property of a stochastic system that every state can be reached out of any other state, is quite simple. In the SIM simulation, each configuration consists of closed loops of occupied links. The worm algorithm alters such a configuration by adding, modifying or removing such a loop. As the worm algorithm can affect any link, there is no closed loop it cannot form, delete, or change in any way. Hence, starting from any initial configuration, any other configuration can be created using one or several applications of the worm algorithm. Thus, the

algorithm is ergodic. In the RBIM simulation, the proof proceeds analogously, even though the configurations are no longer just closed loops but also open chains. However, due to the initial random selection of the antiferromagnetic bonds, every configuration containing open chains can be realized, either directly through selection of the corresponding links in this initial random step or indirectly through modification of an appropriate selection of links by the worm algorithm.
The proof of detailed balance is a bit more involved. In order to fulfil detailed balance, every step of the algorithm must abide to the condition

$$p(C_i)\cdot w_{i\to j} = p(C_j)\cdot w_{j\to i}, \tag{3.2.1}$$

where $p(C_i)$ denotes the probability to be in the configuration C_i and $w_{i\to j}$ stands for the probability to produce the configuration C_j out of C_i in the next step of the algorithm. These steps can be divided into different categories. For both worm types, they are (a) worm propagation onto a non-erroneous link, (b) worm propagation onto an erroneous link as well as (c) start and termination of worm propagation. The categories are also depicted in Figure 3.2.1 for the WHO-version of the worm algorithm and in Figure 3.2.2 for the WHAT-version. From these figures it can easily be determined for all of these categories expect WHO(c) that the steps belonging to them obey detailed balance.
For WHO(c), this proof is a bit more sophisticated, since here the worm can jump from one position to another once its two ends meet and the new initiation site gets selected randomly. This occurs with probability ω and each site has the same probability $\frac{1}{V}$ to be selected. However, in that form the algorithm would violate detailed balance. To correct for this, the weight of the configurations consisting only of closed loops, C_2 and C_3 in Figure 3.2.1, needs to be adapted. They thus receive an additional weight factor X. Because in this algorithm, only the configurations with closed loops get counted, because to each of these loops the same weight factor is multiplied and because in the end only the proportion between the sums of such configuration weights form the result, this does not change the outcome of the algorithm.
There is, however, one case where detailed balance cannot be achieved with this modification: By fulfilling detailed balance X and ω are set into a relation. It is given by

$$\omega = 1 - \frac{1}{X}. \tag{3.2.2}$$

This means that the value of X directly depends on the jump probability ω. For $\omega \to 1$, X thus approaches $X \to \infty$. At this point, the algorithm breaks down, because the worm no longer propagates but just jumps from one site onto another, so that ergodicity gets violated. Hence, for ω a value strictly smaller than 1 needs to be chosen. There are, however, no further constraints on this value. Thus, it was set to $\omega = 0.5$, as it had already been mentioned in the previous subsection.

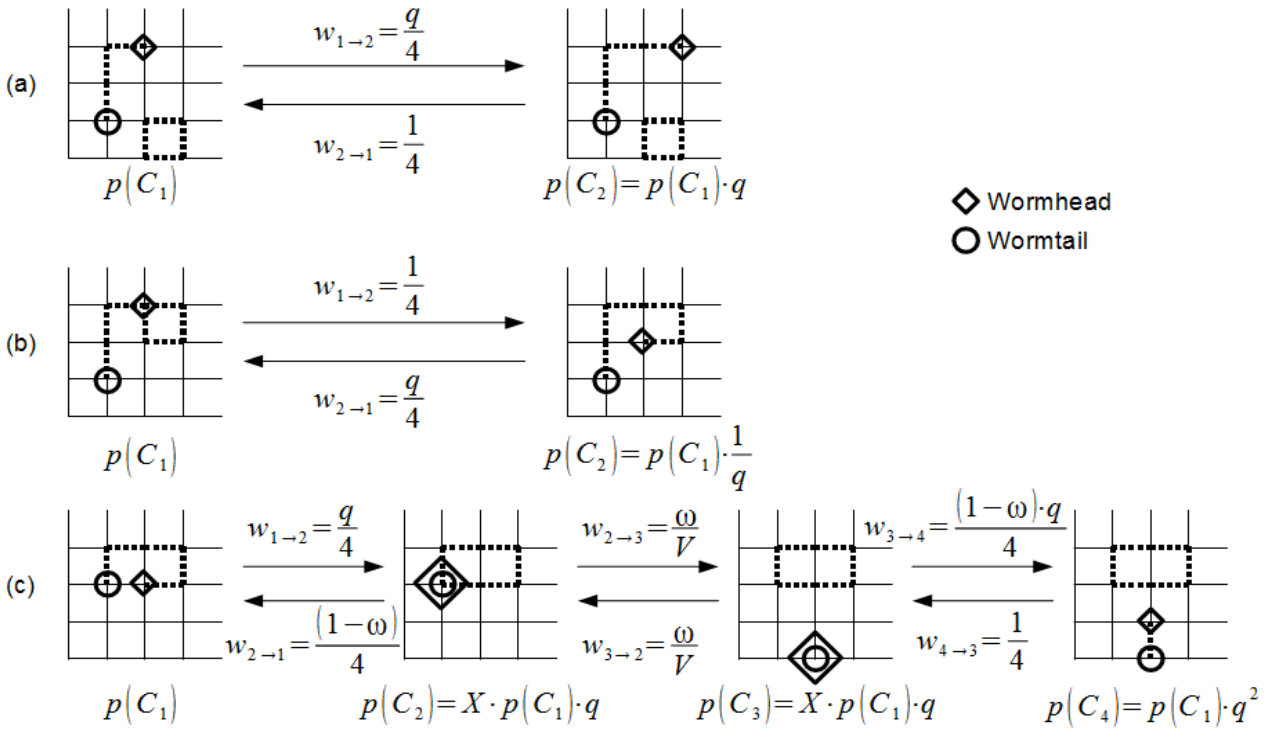

*Figure 3.2.1: Algorithm steps of the **WHO**-version of the worm algorithm. These steps fall into different categories:*
(a) worm propagation onto a non-erroneous link
(b) worm propagation onto an erroneous link
(c) worm propagation termination, worm jumping and worm propagation start
On each lattice, the dotted lines represent occupied links, while the continuous lines represent the unoccupied ones. The circle marks the position of the wormtail, the rhombus the position of the wormhead. The arrows between the lattices represent the steps the algorithm takes from one configuration to the other.
Under each lattice, the weight of its configuration $p(C_i)$ is shown. It is presented relative to the weight of the first configuration on the left $p(C_1)$. Above respectively below each arrow, the probability $w_{i\to j}$ that this step will occur is depicted. The parameters used in the formulae are: the link switch probability q, the worm jump probability ω, the number of sites on the lattice V and the additional weight factor X for configurations that only consist of closed loops.

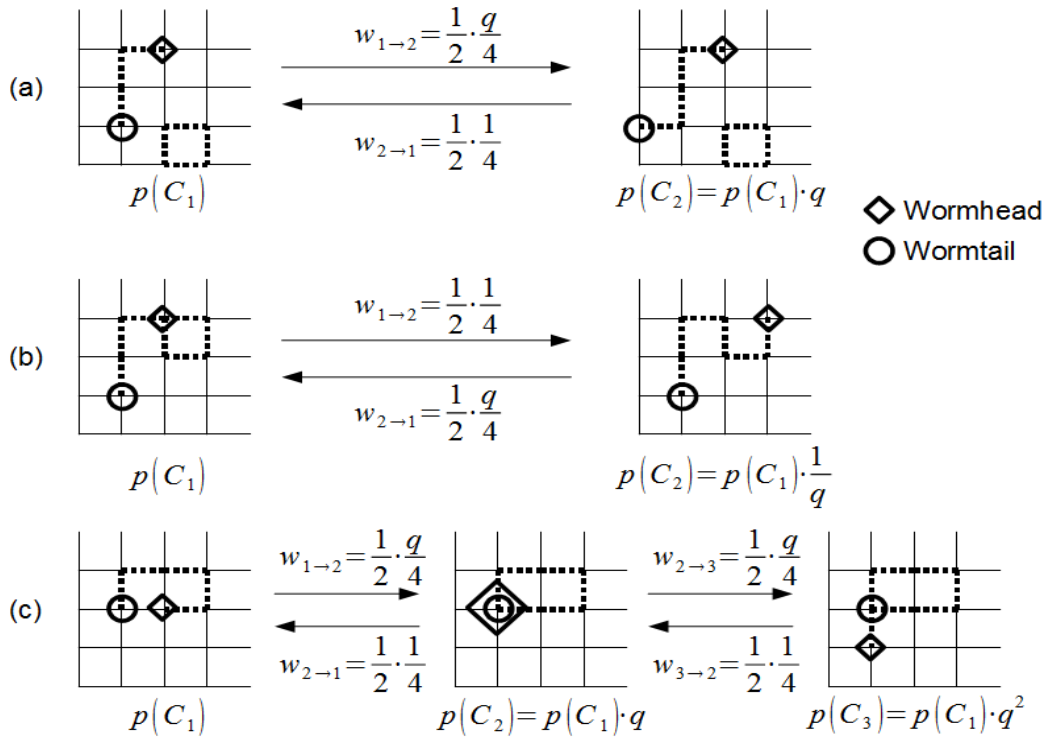

*Figure 3.2.2: Algorithm steps of the **WHAT**-version of the worm algorithm. These steps fall into different categories:*
(a) worm propagation onto a non-erroneous link
(b) worm propagation onto an erroneous link
(c) worm propagation termination and start
On each lattice, the dotted lines represent occupied links, while the continuous lines represent the unoccupied ones. The circle marks the position of the wormtail, the rhombus the position of the wormhead. The arrows between the lattices represent the steps the algorithm takes from one configuration to the other.
Under each lattice, the weight of its configuration $p(C_i)$ is shown. It is presented relative to the weight of the first configuration on the left $p(C_1)$. Above respectively below each arrow, the probability $w_{i\to j}$ that this step will occur is depicted. The parameters used in the formulae are: the link switch probability q, the worm jump probability ω, the number of sites on the lattice V and the additional weight factor X for configurations that only consist of closed loops.

3.3. Performed calculations

After the two variants of the worm algorithm had been developed, they were run simulating the SIM in order to analyse how fast these simulations thermalize, i.e. how long it takes for them to go from the initial configuration into one that is not autocorrelated to it anymore and where thus the distribution of created configurations is no longer biased by the selective constraints on the initial configuration. These runs were performed on four different lattices with the following properties: The lattices had a size of 2x3 (this means the lattice has a horizontal length of 2 and a vertical length of 3), 3x2, 1x1 respectively 2x2. All used a link switch probability of $q=0.25$. The probability measure to switch the link states along a plaquette in order to create a random initial lattice configuration of the trivial homology class was set to $p(iPS)=0.4$ for all lattices and for each of them the simulation was performed several times with a different number of worm algorithm applications ranging from 10^1 to 10^7 in powers of ten. For all of these simulations, it was examined how the proportion $P_{trivial}$ of created configurations belonging to the trivial homology class developed with an increasing number of data points created in the simulation. The value of $P_{trivial}$ at the different stages during this development was compared to that at the end where it had converged towards a stable value. The analysis of this development showed that already after $1\cdot 10^5$ applications of the worm algorithm, the difference between the current $P_{trivial}$ and the

final one was in the order of per milles. After $5 \cdot 10^5$ to $2 \cdot 10^6$ applications of the worm algorithm (depending on the lattice size), the difference decreased to the order of a tenth of a per mille, so that from here on the first three positions after the decimal point of $P_{trivial}$ did not change as new data points were added. Since this accuracy was good enough for the purpose of this thesis, it was decided to not use a so called thermalization phase of worm algorithm applications without data collection in the beginning of the simulation, in which the above mentioned autocorrelation gets eliminated. Instead, it was determined that if the simulation consists of 10^7 worm algorithm applications of which each time the configuration gets measured, the effects of the autocorrelation become negligibly small. The so obtained value is thus accurate enough to be used further. Furthermore, it was determined that if less accuracy is needed (e.g. to get an overview on how $P_{trivial}$ behaves as a function of q in a certain region) a simulation with 10^5 worm algorithm applications without thermalization phase suffices. In addition, the omission of the thermalization phase significantly shortened the time needed to perform the simulations and thus made it possible to perform more simulations, especially on larger lattices.

After this, a large variety of tests was performed on the two variants of the worm algorithm. This was done to ensure that they worked properly. Since this simulation not only revealed the behaviour of the SIM, but also plays its part in the simulation of the RBIM, the correct performance of the latter simulation was also endorsed by these tests.

The first of these tests was performed to check whether the correct results are produced and whether by increasing the number of times the algorithm is run the accuracy of the result increases. For this, the values of q and $p(iPS)$ did not yet matter much and were thus selected randomly to $q = 0.4$ and $p(iPS) = 0.55$. A small lattice size of 2x4 was used. This keeps the number of different configurations low. Thus, apart from the result determined by the worm algorithms, the accurate result is easily calculated by determining and adding up the weights of the individual configurations, a method referred to as *brute force*. By this it can be checked how well the worm algorithm performs and the presence of coding errors can be detected. Furthermore, the number of times the worm algorithm was applied was varied between the individual test runs. It ranged in powers of ten from 10^1 to 10^7.

In the next test, it was checked that different values of the probability measure to switch the link states along a plaquette in order to create a random initial lattice configuration of the trivial homology class $p(iPS)$ do not have an effect on the results and do not produce any systematic error. For this, a small lattice of size 3x3 was used on which the link switch probability q was set to 0.2. For both versions of the worm algorithm several test runs were performed with different values of $p(iPS)$. These ranged from 0 to 1 in steps of 0.05. Each test run consisted of 10^7 applications of the worm algorithm. The results were compared to the accurate results obtained by a brute force algorithm.

The final test was the most elaborate. Instead of just checking the behaviour of a certain property at some randomly chosen conditions, this test systematically went through many different small lattice sizes and link switch probabilities to check the accuracy of the results. To be more precise, for all the lattice sizes that were considered small enough so that an accurate result could be obtained using a brute force algorithm, i.e. the lattice sizes 1x1, 1x2, 2x1, 1x3, 2x2, 3x1, 1x4, 2x3, 3x2, 4x1, 1x5, 2x4, 3x3, 4x2, 5x1, 1x6, 2x5, 3x4, 4x3, 5x2, 6x1, and for all link switch probabilities q from 0 to 1 in steps of 0.05, a test run was performed. Each such test run consisted of 10^7 worm algorithm applications and $p(iPS)$ was each time set to be 0.25. All the obtained results were finally compared to the accurate results from the brute force algorithm.

The next phase of testing was started after all the aforementioned tests were performed successfully, i.e. the distribution of the obtained results fit to a Gaussian distribution around the accurate result with respect to the obtained standard deviations. The exceptions in this regard were the cases where q was close to either 0 or 1. In these cases, for reasons that have not been determined, the results rarely matched well to the accurate values. However, these cases occur on the extremal parts of the

interval of values q can take. The crossing from one phase to the other, which is examined in this thesis, on the other hand, is not expected to occur in these peripheral regions but more in the center of the interval. Thus, these exceptions are of little relevance in regard to the goal of this thesis. Therefore, the algorithm was not adapted to get rid of this wrong behaviour. Instead it was left unchanged and whenever such an outlier appears in the results, it will just be pointed out. However, for all of the remaining values of the link switch probability q that were analysed, the tests reliably produced results that only varied from the accurate result in a manner typical for stochastic errors.

In this next phase of testing, the focus was set on scaling. Test runs were performed on lattices of larger size, namely 100x100 and 1000x1000. The link switch probability q was set to 0.4 and $p(iPS)$ to 0.55. In each test run the number of times the worm algorithm was applied changed. Again, it ranged in powers of ten from 10^1 to 10^7. Due to the far too large number of different configurations which can be realized on these lattice sizes, an accurate result could not be obtained by a brute force algorithm. Thus, to check whether the algorithm still worked properly in these larger systems, the results obtained from the different test runs with a different number of worm algorithm applications were compared to each other. It was determined whether their results converge towards certain values as the number of worm applications increases and how well the differences in these measurements with increasing accuracy correlate to their errors.

The success in this test together with that in the former ones lead to the conclusion that the two variants of the worm algorithm can be considered to reliably produce accurate results. Hence, the actual numerical calculations could be started. These were performed with both variants of the worm algorithm on lattices of five different sizes, namely 20x20, 40x40, 60x60, 80x80 and 100x100. In all runs $p(iPS)$ was put to 0.55.

First, the SIM was simulated to check that the algorithms are able to correctly reproduce the spin change behaviour of the two Ising models. Therefore, the following calculations were performed:

The first series of runs was used to get an overview on how the probability $P_{trivial}$, that a produced configuration belongs to the trivial homology class, depends on the link switch probability q. Therefore, in this series both variants of the algorithm were run for every q from 0 to 1 in steps of 0.01. As this data was only used for an overview of the dependence, it did not need to be very accurate. Thus, the worm algorithm was only applied 10^5 times in each run. The results are presented in Section 4.1.

In the second series of runs, the attention was directed to the interval where the phase crossing had occurred in the data of the overview series of runs. Here, both variants of the algorithm were run for every q from 0.350 to 0.450 in steps of 0.001. This was done to produce a finer view of this interval. Thus, the accuracy was not increased with respect to the previous runs, so that each run still consisted of 10^5 applications of the worm algorithm. The results of this series of runs (to be called *zoom-in at 0.350<=q<=0.450*) are discussed in Section 4.2. They allowed the refinement of the limits for the location of the critical link switch probability q_{crit}, which marks the position of the phase transition on the infinite size lattice and is contained inside the region of phase crossing in the finite size lattice, to be set to $0.405 \leq q_{crit} \leq 0.420$.

Thereafter, in this new interval, high accuracy measurements were performed. For every q between 0.405 and 0.420 in steps of 0.001 both variants of the algorithm were run. In each run, the worm algorithm was applied 10^7 times. The results of these series of runs (to be called *zoom-in at 0.405<=q<=0.420*) are presented in Section 4.3.

In the final series of runs of the SIM simulation (to be called *zoom-in at 0.4130<=q<=0.4150*) the interval of q between 0.4130 and 0.4150 was analysed. This interval was chosen, since due to the findings of the former series of runs, the boundaries for the measured q_{crit} could be narrowed down to this interval. Both variants of the algorithm were run in steps of 0.0001 on a 1000x1000-lattice. The large size of the lattice was chosen to further narrow down the interval in which the phase crossing takes place and inside which q_{crit} is located. Because of the long

calculation time on lattices of this size, in each run the worm algorithm was only applied 10^5 times. The results are presented in Section 4.4.

For the simulation of the RBIM only two series of runs were performed:

In the first series of runs (to be called *RBIM overview*) a coarse overview of the interval, in which the phase crossing was expected according to Dennis et al. [2], was performed. For this purpose, on a 100x100-lattice both variants of the algorithm simulating the RBIM (the one which incorporated the WHO- as well as the one which incorporated the WHAT-worm algorithm) were run for every p_{AFB} from 0.09 to 0.13 in steps of 0.01. For each run, 10^3 different initial antiferromagnetic bond patterns were created and for each of these, the worm algorithm was applied 10^4 times. This resulted in a total of 10^7 data points per run. The results are presented in Section 5.1.

Based on the results of the previous series of runs, the phase crossing was confirmed to lie in the interval $p_{AFB,crit} \in [0.09, 0.12]$. Hence, a more detailed analysis of this interval was performed in the next series of runs (to be called *zoom-in at 0.090<=p_AFB<=0.120*). Both RBIM simulation algorithms were applied on lattices of the sizes 20x20, 40x40, 60x60, 80x80 and 100x100. They were run with values of p_{AFB} between 0.09 and 0.12 in steps of 0.001. As in the previous series of runs, for each run 10^3 different initial antiferromagnetic bond patterns were created and for each of these, the worm algorithm was applied 10^4 times. The results are presented in Section 5.2.

4. Standard Ising model

This section presents the results of the numerical simulation of the standard Ising model (SIM) and discusses them. In this model, the worm algorithms were checked to reproduce the correct spin change behaviour before applying them to simulate the random-bond Ising model. The series of runs of the simulation presented in this section have been described in detail in Section 3.3.

First, the development of the proportion of created configurations that belong to the trivial homology class $P_{trivial}$ is presented as a function of the link switch probability q in Subsection 4.1 for the whole range of possible values of q, i.e. $q \in [0, 1]$.

Thereafter, the behaviour of $P_{trivial}$ is shown in more detail in the region of q, in which the phase crossing was determined to occur in this first series of runs. Thus, in Subsection 4.2 the behaviour of $P_{trivial}$ as a function of q is examined in more detail inside the interval $q \in [0.350, 0.450]$.

The results of this second series of runs narrowed down the location of the critical link switch probability q_{crit} into the interval $q \in [0.405, 0.420]$. This interval was analysed in the third series of runs of which the results are shown in Subsection 4.3 and evaluated in Subsection 4.5. It provided a more accurate picture of the behaviour of $P_{trivial}$ around q_{crit}.

The evaluation of these results allowed further narrowing down the location of q_{crit} into the interval $q \in [0.4130, 0.4150]$. This interval is examined in the fourth and final series of runs of the SIM simulation on a 1000x1000 lattice in Subsection 4.4. The results are then compared to those of the previous series of runs in Subsection 4.5.

4.1. Overview

In the first series of runs of the simulation of the SIM, the two variants of the algorithm were used to produce a whole-scale overview on how the proportion $P_{trivial}$ of produced configurations belonging to the trivial homology class develops in dependence of the link switch probability q. The results are illustrated in Figure 4.1.1 for the simulation using the WHO-variant of the worm algorithm and in Figure 4.1.2 for the simulation using the WHAT-variant.

In these figures, it can be seen that both variants of the algorithm produce a similar outcome. A more detailed comparison between the results from these two variants of the algorithm is performed in Section 6. In both cases, $P_{trivial}$ plateaus at 1 for small q, then decreases rapidly around $q=0.4$ and eventually plateaus again for larger q. This time at 0.25. It can furthermore be seen that the decrease becomes steeper as the size of the lattice increases. This also extends the lengths of the two plateaux. This leads to the conclusion that the critical link switch probability q_{crit} lies in this region of q where the phase crossing occurs, namely inside the interval $q \in [0.350, 0.450]$. This finding has been used to determine the boundaries of the interval, at which to focus in the next series of runs.

In both variants of the algorithm, outliers can be seen for $q=1.0$. Outliers at this value of q were already present throughout the tests of the algorithms, indicating that the used method breaks down in this region. The reason for this has not been further investigated, since this break down occurs in the periphery of the link switch probability q range, that means far off the region where the phase crossing, onto which the focus of this thesis is set, is expected to occur.

In the WHAT-variant, there are two further outliers in the data set of the 60x60 lattice for $q=0.01$ and $q=0.03$. Their value of $P_{trivial}$ is 4σ lower than expected in this plateau region, where $P_{trivial}$ should stay approximately at 1. As with the outliers at $q=1.0$, these ones are also located in one of the regions where the algorithm breaks down. Since these outliers are also located far off the region where the phase crossing occurs, they are as well seen to have little relevance.

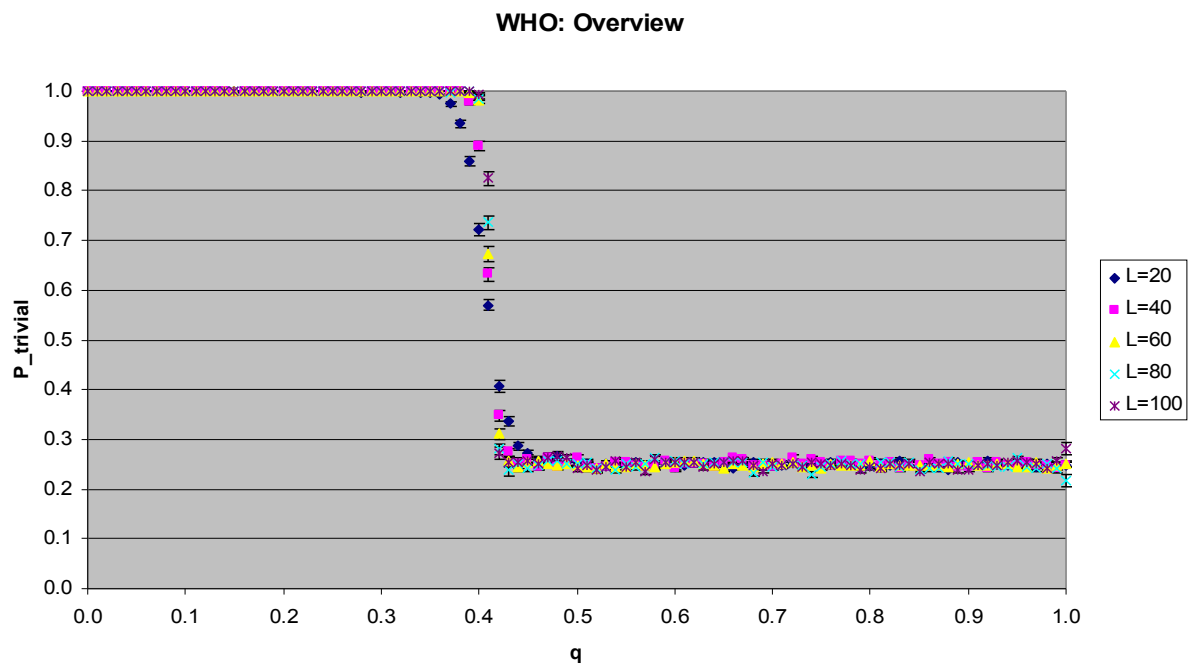

Figure 4.1.1: Diagram illustrating the results of the numerical calculations of the "overview" series of runs performed using the "wormhead only" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results give an overview on how $P_{trivial}$ behaves as a function of q along the whole scale.

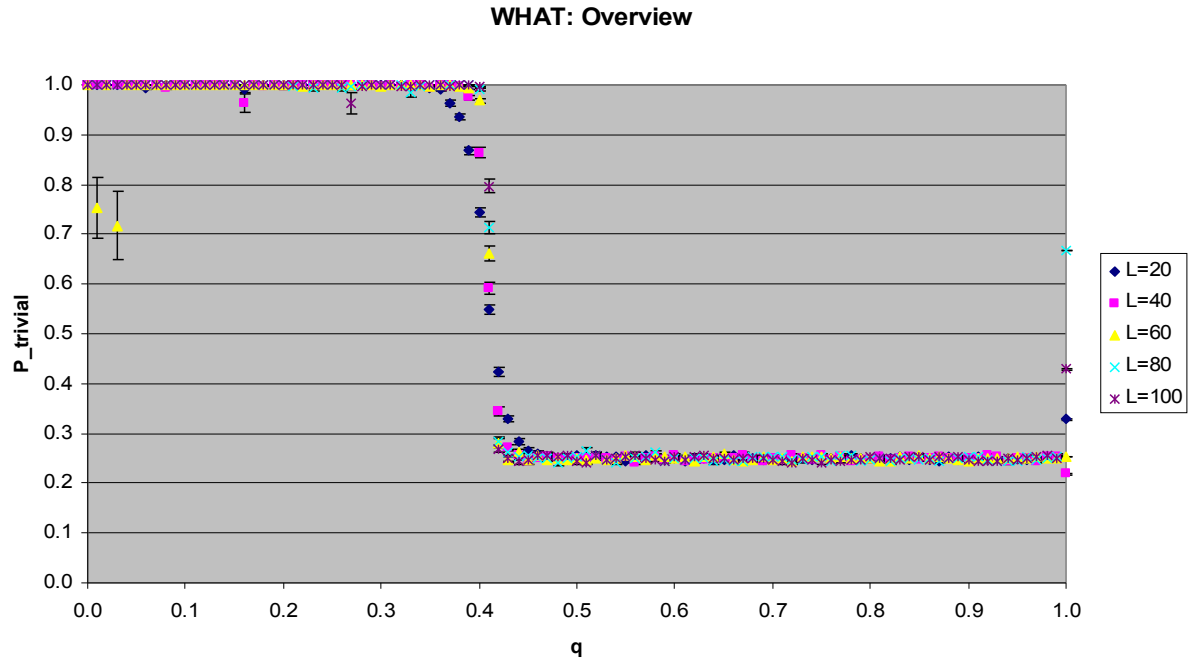

Figure 4.1.2: Diagram illustrating the results of the numerical calculations of the "overview" series of runs performed using the "wormhead and -tail" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results give an overview on how $P_{trivial}$ behaves as a function of q along the whole scale.

4.2. Zoom-in on interval around q_{crit}

In the second series of runs of the simulation of the SIM, the two variants of the algorithm were used to zoom in on the development of the proportion $P_{trivial}$ of produced configurations belonging to the trivial homology class in the interval in which the phase crossing associated to the critical link switch probability q_{crit} had been observed in the overview series of runs. The boundaries of the interval were set generously to ensure that q_{crit} lies between them. Thus, it reaches from $q=0.350$ to $q=0.450$. The results of the simulations using the WHO-variant of the worm algorithm are shown in Figure 4.2.1, those of the simulations using the WHAT-variant in Figure 4.2.2.

Both variants of the algorithm produced very similar results. Their comparison is performed in Section 6. The produced results correspond with the expectations from the results of the overview series. In both diagrams, on the left $P_{trivial}$ forms a plateau at the value 1. This is, in both diagrams, followed by a drastic decrease of $P_{trivial}$ around $q=0.41$, which becomes steeper with increasing lattice size. Finally, the value of $P_{trivial}$ again stabilizes at around 0.25 for all the graphs in both diagrams.

In both diagrams, the decrease of $P_{trivial}$ on the 100x100-lattice starts at $q=0.405$ and ends at $q=0.420$. Since the plateaux in front respectively behind these points represent the different phases the system can be in, q_{crit} must lie between them. This finding has been used to determine the boundaries of the interval, at which to focus in the next series of runs.

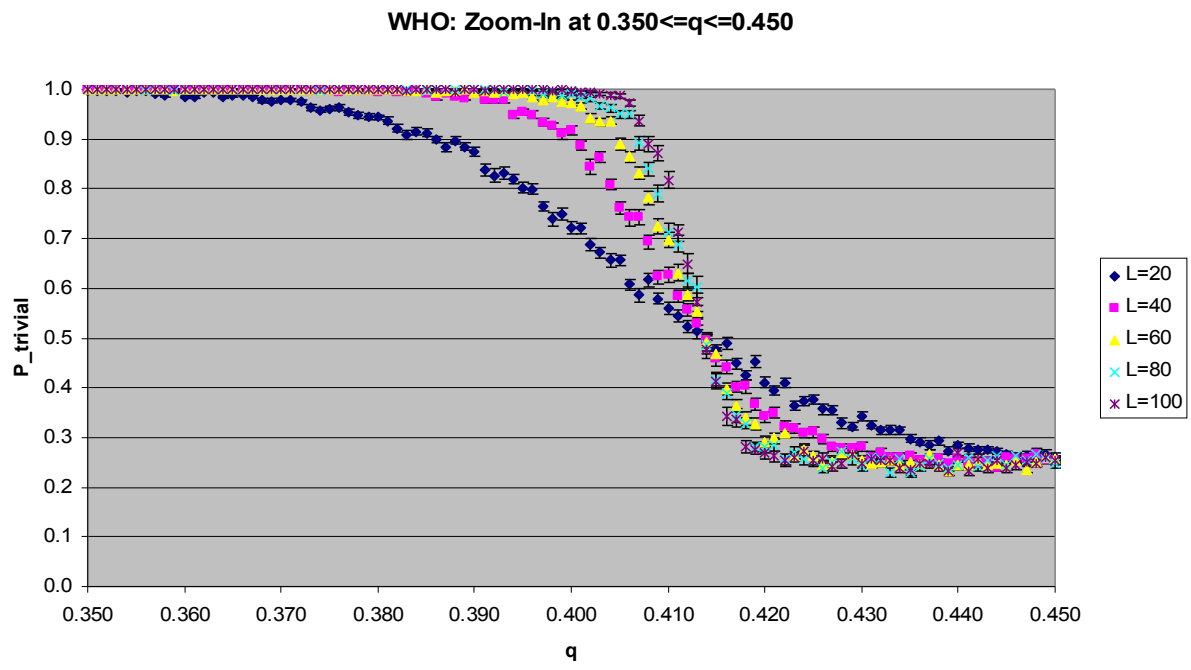

Figure 4.2.1: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.350<=q<=0.450" series of runs performed using the "wormhead only" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.350$ and $q=0.450$.

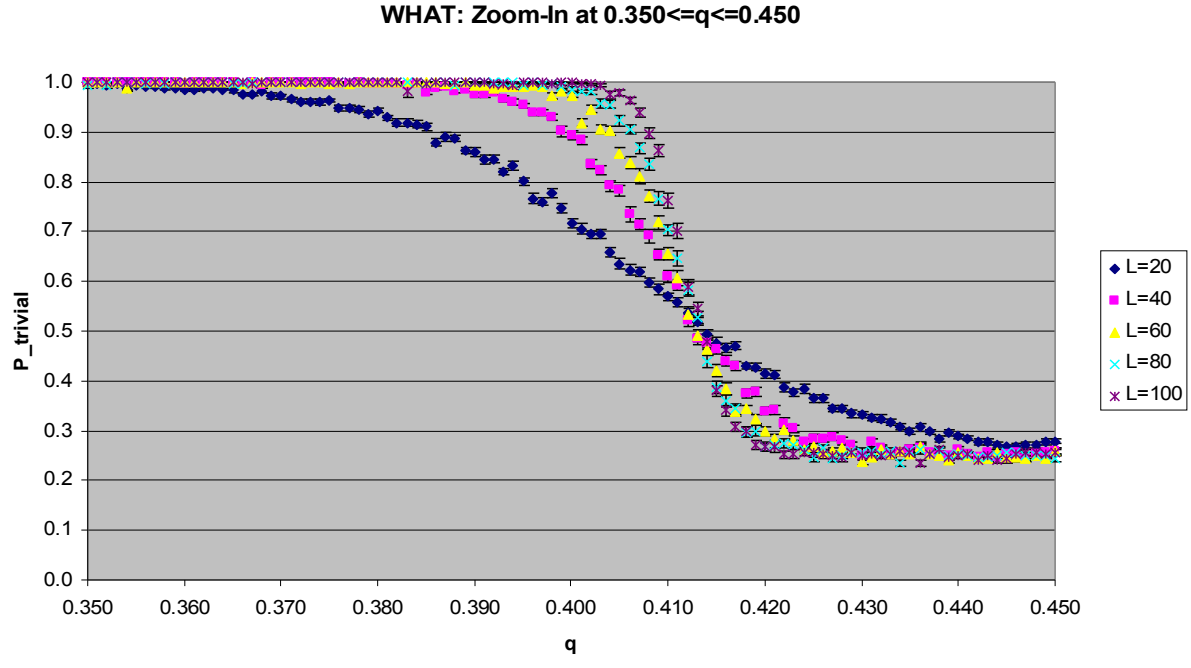

Figure 4.2.2: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.350<=q<=0.450" series of runs performed using the "wormhead and -tail" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.350$ and $q=0.450$.

4.3. More accurate values in interval around q_{crit}

In the third series of runs of the simulation of the SIM, the two variants of the algorithm were again used to determine the values of the proportion $P_{trivial}$ of produced configurations belonging to the trivial homology class in the interval of the link switch probability q, in which the phase change associated to the critical link switch probability q_{crit} had been observed in the overview series of runs. This time, however, the boundaries of the interval were narrowed down due to the findings from the previous series of runs on where to locate q_{crit}. The lower boundary was set at $q = 0.405$ and the upper at $q = 0.420$. Furthermore, the accuracy of the produced values of $P_{trivial}$ was increased to 10^7 applications of the worm algorithm. The results of the simulations using the WHO-variant of the worm algorithm are presented in Figure 4.3.1, those of the simulations using the WHAT-variant in Figure 4.3.2.

Both variants of the algorithm produce very similar results. Their detailed comparison with respect to the different variants of the algorithm is performed in Section 6. As expected from the setting of the boundaries, both diagrams show the decrease in $P_{trivial}$ that occurs around q_{crit}. Again, this decrease gets steeper the larger the lattice becomes. Furthermore, at the point $q = 0.414 \pm 0.001$ (WHO) respectively $q = 0.413 \pm 0.001$ (WHAT) the graphs from the different lattice sizes intersect each other.

The analysis of these results will be done sophisticatedly and thus performed separately in Subsection 4.5.

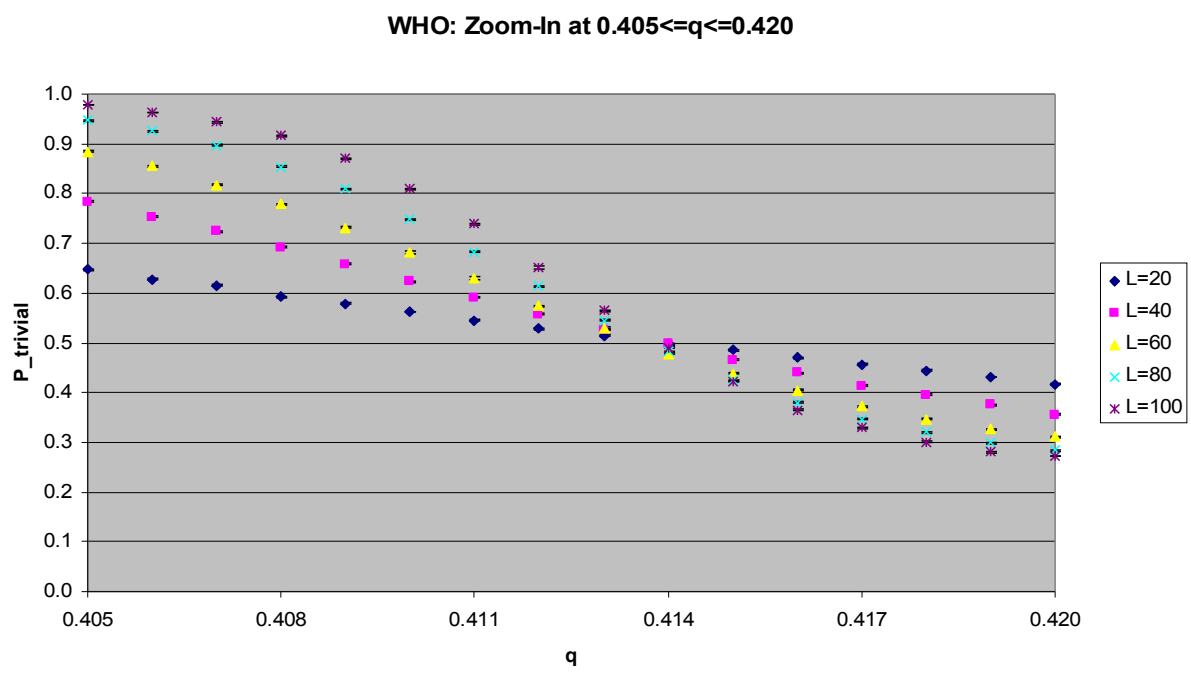

Figure 4.3.1: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.405<=q<=0.420" series of runs performed using the "wormhead only" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.405$ and $q=0.420$. Note the small size of the error bars due to the high accuracy of the results due to the large number of worm algorithm applications.

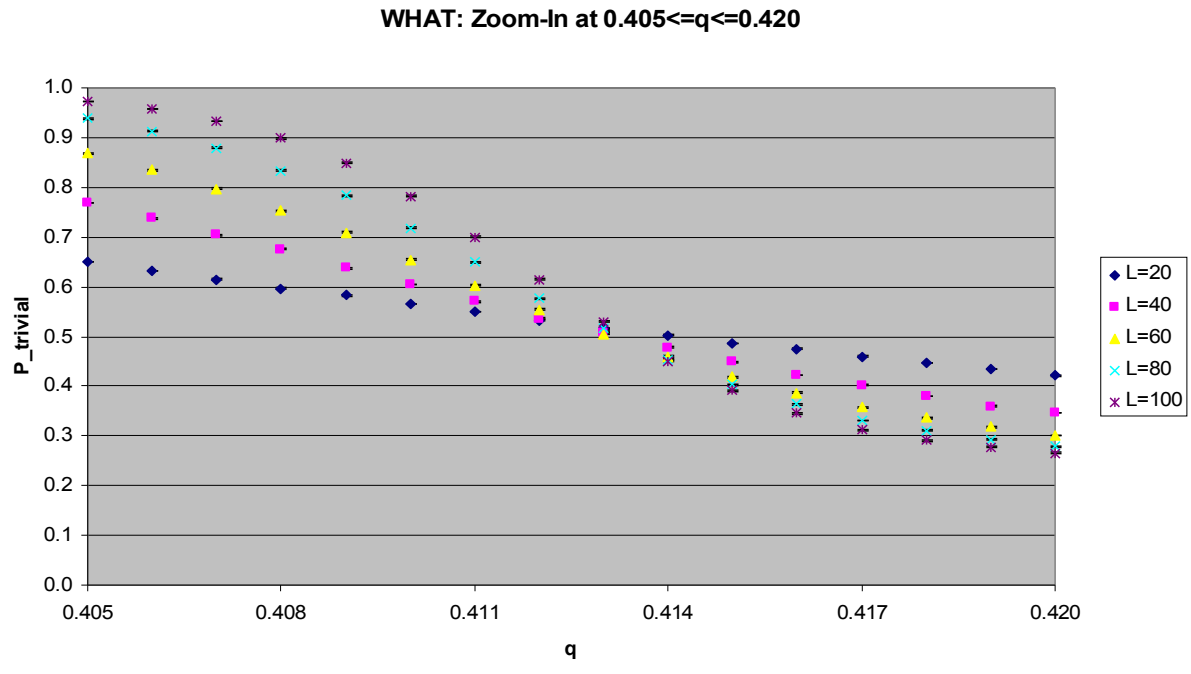

Figure 4.3.2: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.405<=q<=0.420" series of runs performed using the "wormhead and -tail" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.405$ and $q=0.420$. Note the small size of the error bars due to the high accuracy of the results due to the large number of worm algorithm applications.

4.4. Zoom-in on interval around q_{crit} on a 1000x1000-lattice

In the fourth series of runs of the simulation of the SIM, the two variants of the algorithm were used to determine the values of the proportion $P_{trivial}$ of produced configurations belonging to the trivial homology class inside the link switch probability q interval of $q \in [0.4130, 0.4150]$. It was inside this interval that, according to the findings of the previous series of runs, the critical link switch probability q_{crit} should lie. To better see the phase crossing in this narrow interval, the lattice size was increased to 1000x1000. This, however, came at the cost of reducing the accuracy of each run to 10^5 applications of the worm algorithm. The results of the simulations using the WHO-variant of the worm algorithm are shown in Figure 4.4.1, those of the simulations using the WHAT-variant in Figure 4.4.2.

As it can be seen in these figures, both variants of the algorithm produce very similar results. The detailed comparison of the results between the two variants is performed in Section 6. In the diagrams it can be seen that $P_{trivial}$ plateaus around 1 for the lower q. At $q = 0.4135$, the value of $P_{trivial}$ starts to decrease. This decrease continues until $q = 0.4150$ (WHO), where the diagram ends, respectively till $q = 0.4149$ (WHAT), where a new plateau around $P_{trivial} = 0.25$ starts to form. Such a plateau at the right end of q cannot be seen in the diagram of the WHO-variant of the algorithm. This, however, is likely just because the series terminates before such a plateau could establish itself. These results indicate that the phase crossing on lattices of this size likely completes at a q quite close to 0.4150.

This shows that the critical link switch probability q_{crit} lies inside of the interval $q \in [0.4135, 0.4150]$, a result that goes along with the findings from the previous series of runs simulating this model. However, it does not narrow down much further the boundaries of the interval in which q_{crit} lies.

This result is compared to the one from the previous series of runs in Subsection 4.5.

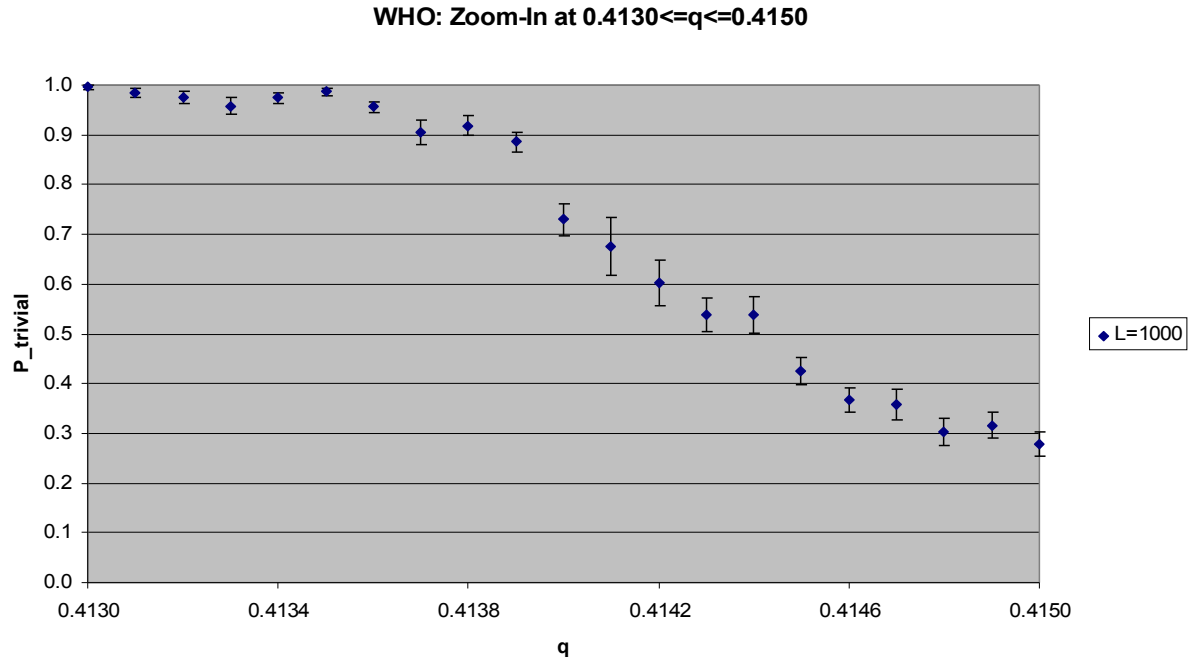

Figure 4.4.1: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.4130<=q<=0.4150" series of runs performed using the "wormhead only" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. The data set was created using a square lattice with a side length of 1000 links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.4130$ and $q=0.4150$.

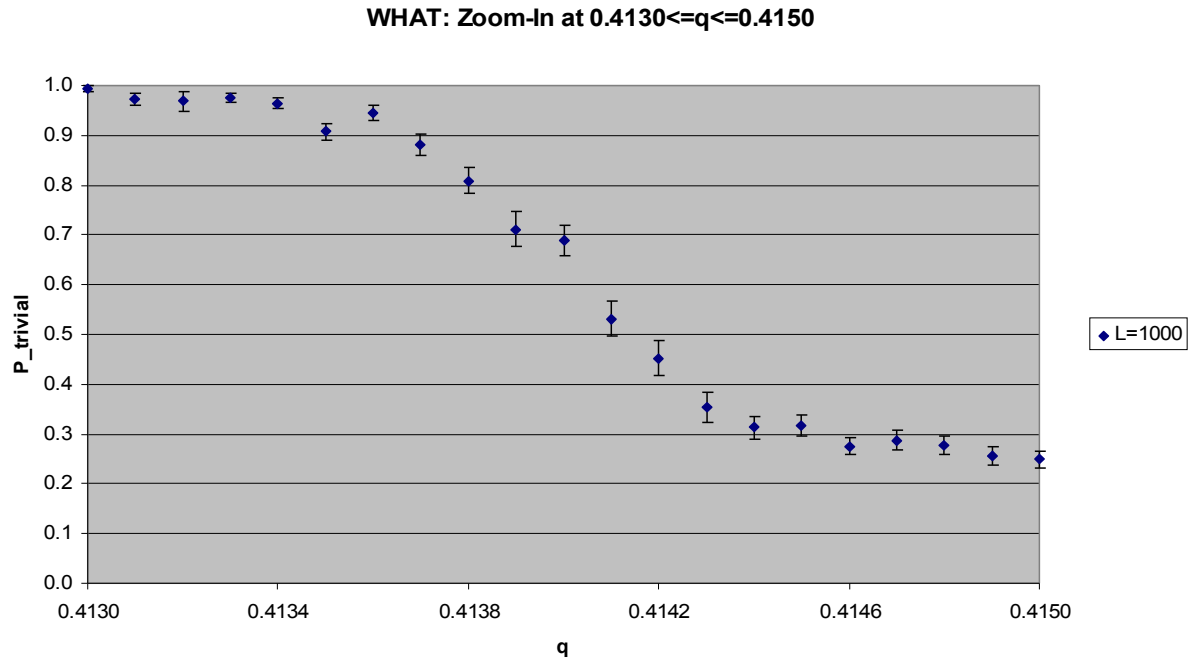

Figure 4.4.2: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.4130<=q<=0.4150" series of runs performed using the "wormhead and -tail" variant of the worm algorithm in the simulation of the SIM. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. The data set was created using a square lattice with a side length of 1000 links.
These results show how $P_{trivial}$ behaves as a function of q in the interval between $q=0.4130$ and $q=0.4150$.

4.5. Analysis of the results

In the series of runs named "zoom-in at 0.405<=q<=0.420" the method applied in the previous series of runs to narrow down the interval, in which the critical link switch probability q_{crit} is located, by using the start and the end of the phase crossing no longer works. This is, because this series of runs was performed on lattices of the same sizes as in the previous ones. This means that the length of the phase crossing stays the same, only the positions of its boundaries will be determine more accurately. While in the previous series of runs this refinement still yielded a significant reduction in the interval size, i.e. the size of the shift of the boundary was one order of magnitude smaller than the interval, in the "zoom-in at 0.405<=q<=0.420" series of runs the refinement is too small to significantly change the boundaries of the interval. Hence, another method needs to be used:

With increasing lattice size, the graphs of $P_{trivial}(q)$ decrease more steeply during the crossover. By this, more points in the vicinity of the critical point q_{crit} move towards a value of $P_{trivial}$ that corresponds to that of the phase they will be in on the infinite-size lattice. For values below q_{crit} this is 1, for values above q_{crit}, this is 0.25. Hence for lattices of different sizes, their graphs of $P_{trivial}(q)$ will intersect close to q_{crit}. Therefore, q_{crit} can be determined by analysing these intersections. In Figure 4.3.1 and Figure 4.3.2 it can easily be seen that the intersections are situated near $q=0.414$ (WHO) respectively $q=0.413$ (WHAT). Around these points, the data points of each lattice size appear to be arranged almost linearly. Thus, the graphs of $P_{trivial}(q)$ in this region can be well approximated using linear regression (see also Figure 4.5.1 and Figure 4.5.2). From this, the intersections between the individual regression graphs can be calculated. Unfortunately, due to their approximate nature (originating from the approximative nature of the algorithm as well as of the here described method to determine q_{crit}) the regression graphs do not all intersect in one point. Instead, each regression graph separately intersects the other ones. The positions of these intersections are shown in Table 4.5.1. It is assumed that these intersection points distribute stochastically around q_{crit}. Hence, their mean value can be used to determine q_{crit}, together with their standard deviation for its error estimate. For the WHO-variant of the worm algorithm, these are

$$q_{crit,WHO}=0.4140\pm0.0005, \tag{4.5.1}$$

and for the WHAT-variant

$$q_{crit,WHAT}=0.4135\pm0.0006. \tag{4.5.2}$$

Analogously, the value of $P_{trivial}$ at q_{crit} and its error can be determined. For the WHO-variant of the worm algorithm, these are

$$P_{trivial,WHO}\left(q_{crit}\right)=0.490\pm0.025, \tag{4.5.3}$$

and for the WHAT-variant

$$P_{trivial,WHAT}\left(q_{crit}\right)=0.494\pm0.027. \tag{4.5.4}$$

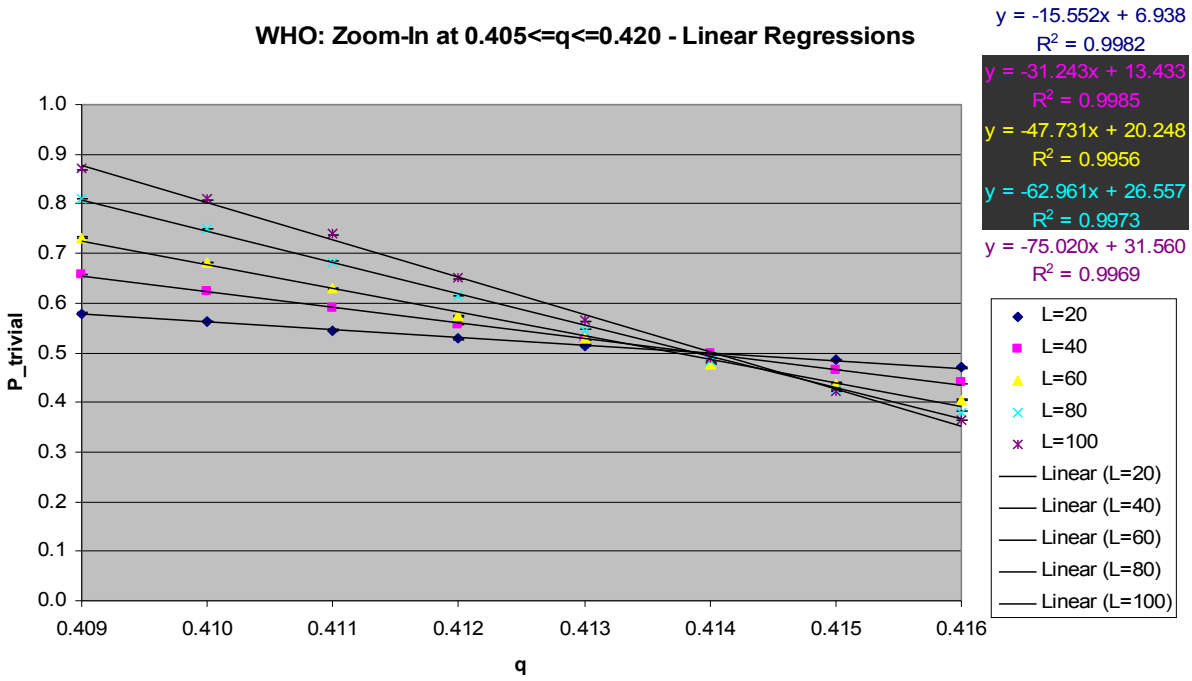

Figure 4.5.1: Diagram showing the linear regression of the result data from the "zoom-in at 0.405<=q<=0.420" series of runs from the simulation of the SIM performed using the "wormhead only" variant of the algorithm. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets are shown. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links. The equations of the regression graphs and their coefficients of determination are listed on the top right. The colours of the equations match those of the individual data sets. Bright colours have a dark background for better readability.
For the regression only the data points between $q=0.409$ and $q=0.416$ were used, because here the behaviour of $P_{trivial}(q)$ was almost linear.

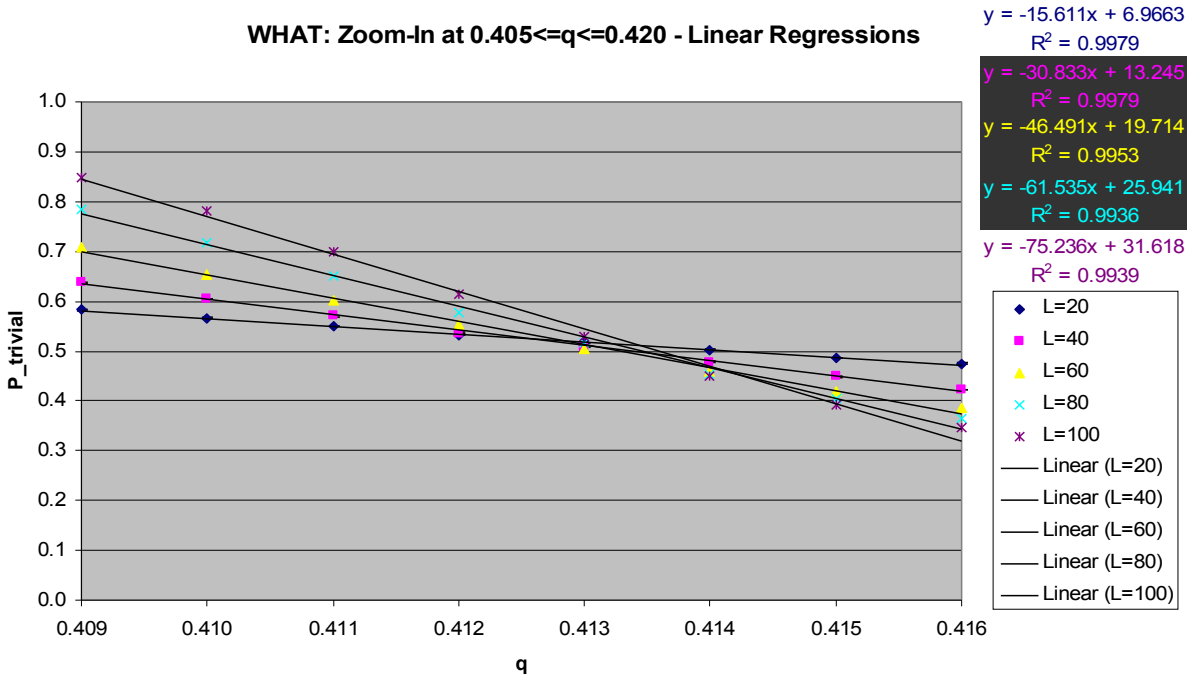

Figure 4.5.2: Diagram showing the linear regression of the result data from the numerical calculations of the "zoom-in at 0.405<=q<=0.420" series of runs from the simulation of the SIM performed using the "wormhead and -tail" variant of the algorithm. The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets are shown. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links. The equations of the regression graphs and their coefficients of determination are listed on the top right. The colours of the equations match those of the individual data sets. Bright colours have a dark background for better readability.
For the regression only the data points between $q=0.409$ and $q=0.416$ were used, because here the behaviour of $P_{trivial}(q)$ was almost linear.

Intersection	WHO		WHAT	
L_1;L_2	q	P_trivial	q	P_trivial
20;40	0.41391	0.50098	0.41245	0.52737
20;60	0.41361	0.50563	0.41282	0.52156
20;80	0.41383	0.50220	0.41319	0.51589
20;100	0.41404	0.49899	0.41344	0.51191
40;60	0.41332	0.51921	0.41319	0.50474
40;80	0.41379	0.50463	0.41355	0.49344
40;100	0.41408	0.49555	0.41378	0.48637
60;80	0.41430	0.47281	0.41393	0.46998
60;100	0.41454	0.46123	0.41411	0.46195
80;100	0.41485	0.43824	0.41430	0.44770

Table 4.5.1: Positions of the intersections between the linear regression graphs of the data points around $q=0.414$ (WHO), respectively $q=0.413$ (WHAT) in the "zoom-in at 0.405<=q<=0.420" series of runs of the simulation of the SIM. The left column indicates the different intersections. The two numbers name the lattice sizes of which the linear regression graphs intersect. This occurs at the coordinates presented in the central and right column doublet for the WHO- respectively WHAT-variant of the algorithm.

The values obtained for q_{crit} in eq. (4.5.1) and eq. (4.5.2) can be compared to the analytically derived actual value for the critical point of the phase transition in the SIM, which has been derived in Section 3.1 to be $q_{crit}=p_{SC,crit}=\sqrt{2}-1\approx 0.41421...$ (see eq. (3.1.18)).

This yields a deviation for $q_{crit,WHO}$ of $0.42\,\sigma$ and for $q_{crit,WH\&T}$ of $1.27\,\sigma$. As the serendipitous finding presented in Section 7 indicates, the exact value of $P_{trivial}$ at q_{crit} likely is $P_{trivial}\left(q_{crit}\right)=0.5$, at least for finite sized lattices. This value can therefore be compared to those obtained from the "zoom-in at 0.405<=q<=0.420" series of runs, which are given in eq. (4.5.3) and eq. (4.5.4). This comparison yields a deviation for $P_{trivial,WHO}\left(q_{crit}\right)$ of $0.41\,\sigma$ and for $P_{trivial,WH\&T}\left(q_{crit}\right)$ of $0.22\,\sigma$.

Another approach to determine q_{crit} is to narrow down the interval, in which the phase crossing takes place and in which q_{crit} is thus located, by increasing the lattice size. This is done in the "zoom-in at 0.4130<=q<=0.4150" series of runs in Section 4.4. The results from this series of runs show that the phase transition occurs between $q\simeq 0.4135$ and $q\simeq 0.4150$. So far this agrees with the values for q_{crit} obtained above and the exact value determined analytically in the end of Section 3.1, since both of these q_{crit} lie inside the interval. This agreement, however, ends once the data points in the interval are being inspected in more detail, as it is done in Figure 4.5.3 and Figure 4.5.4. In these figures, the two $P_{trivial}\left(q_{crit}\right)$ points from the regression analysis of the "zoom-in at 0.405<=q<=0.420" were added to the results from the series of runs in the interval $q\in[0.4130,0.4150]$.

It can be seen in these two figures that the data points from the "zoom-in at 0.4130<=q<=0.4150" series of runs pass aside the data point representing the exact $P_{trivial}\left(q_{crit}\right)$, rather than incorporating it into their development. This discrepancy is larger than $1\,\sigma$ for both versions of the algorithm. However, it remains still smaller than $2\,\sigma$, being even just slightly more than $1\,\sigma$ for the WHAT-variant.

On the other hand, the data points representing the $P_{trivial}\left(q_{crit}\right)$ acquired from the linear regression fit better with those from the series of runs. Although the $P_{trivial}\left(q_{crit}\right)$ data points are actually far away from those of the "zoom-in at 0.4130<=q<=0.4150" series of runs, they also have a large error compared to the other data points due to the different method by which they were determined. Due to this large error, the discrepancy reduces to slightly less than $1\,\sigma$ for the WHO-variant, respectively slightly more than $1\,\sigma$ for the WHAT-variant.

The different values obtained for q_{crit} are difficult to interpret. This is because there are only two

of them for each variant of the algorithm and because the difference between them and towards the exact value mostly is around 1σ. This makes it difficult to decide whether these values distribute normally and thus whether the observed discrepancy between these values is just caused stochastically or whether there is a systematic error source behind it that biases the obtained results. It is, however, likely that the systematic error between the two versions of the worm algorithm (see Section 6) had an influence on this outcome. Anyway, the here described discrepancy between the different values obtained for q_{crit} is of a small magnitude (it is only due to the high accuracy of the calculations that it became visible at all). It has about the same magnitude as the discrepancy between the two variants of the algorithm. Hence, by decreasing the accuracy by one order of magnitude these discrepancies can be made negligibly small, so that all of these approaches for both variants of the algorithm can confirm the value of

$$q_{crit} = 0.414 \pm 0.0005 . \tag{4.5.5}$$

This result is less than 1σ away from the actual value of q_{crit}. Hence, if used at this accuracy, the here tested worm algorithms are adequate to simulate the spin configuration behaviour of the RBIM.

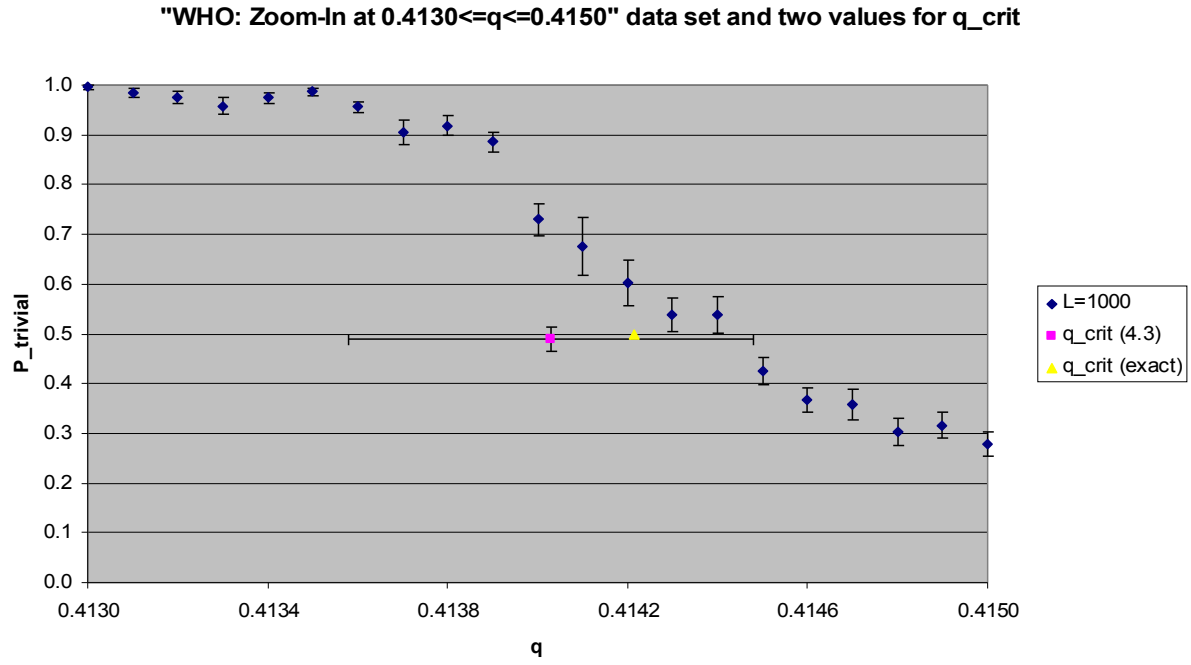

Figure 4.5.3: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.4130<=q<=0.4150" series of runs performed using the "wormhead only" variant of the algorithm simulating the SIM on a 1000x1000 lattice (blue), as well as the value of the critical link switch probability q_{crit} obtained from the linear regression of the results from the "zoom-in at 0.405<=q<=0.420" series of runs (Section 4.3; pink) respectively its exact value (end of Section 3.1; yellow). The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class.
This diagram shows how well the different values obtained for q_{crit} match with the behaviour of $P_{trivial}(q)$ on the 1000x1000 lattice.

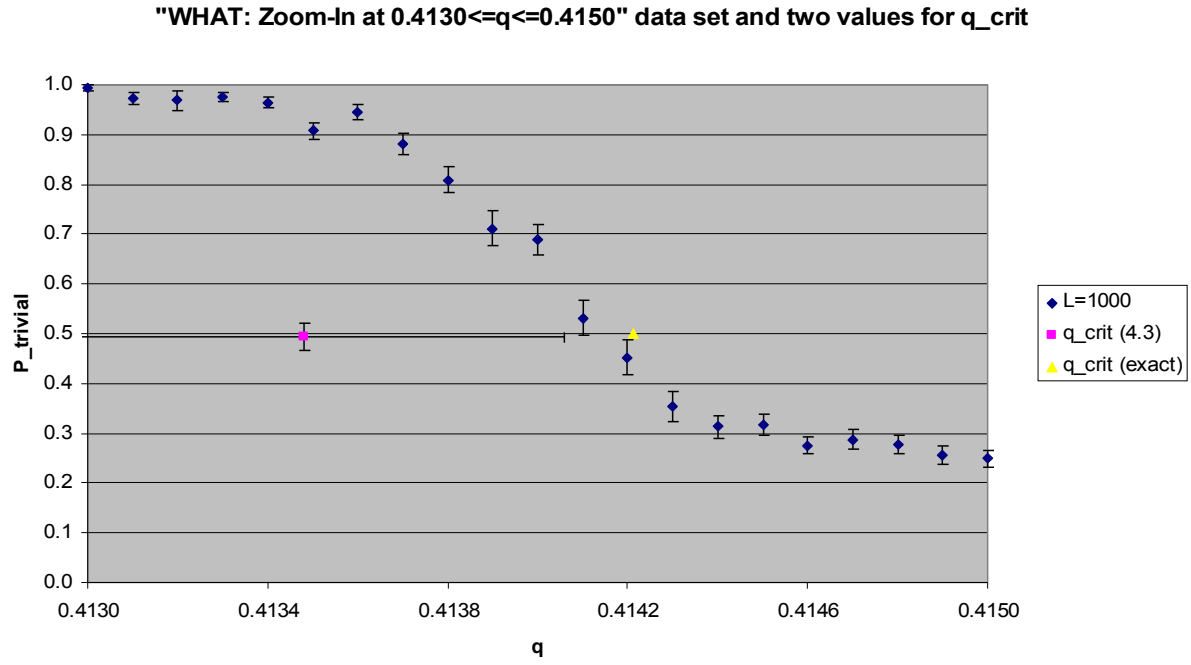

Figure 4.5.4: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.4130<=q<=0.4150" series of runs performed using the "wormhead and -tail" variant of the algorithm simulating the SIM on a 1000x1000 lattice (blue), as well as the value of the critical link switch probability q_{crit} obtained from the linear regression of the results from the "zoom-in at 0.405<=q<=0.420" series of runs (Section 4.3; pink) respectively its exact value (end of Section 3.1; yellow). The horizontal axis represents the link switch probability q. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class.
This diagram shows how well the different values obtained for q_{crit} match with the behaviour of $P_{trivial}(q)$ on the 1000x1000 lattice.

5. Random-bond Ising model

This section presents the results of the numerical simulation of the random-bond Ising model (RBIM) and discusses them. This model is analogous to the error correction model of the toric code. The here determined critical probability $p_{AFB,crit}$ to have an antiferromagnetic bond, where the phase transition occurs for an infinite size lattice, thus corresponds to the accuracy threshold p_{crit} of the toric code error correction model. The simulations of which the results are presented in this section have been described in Section 3.3.

First, in Subsection 5.1 a coarse overview of the behaviour of $P_{trivial}$ as a function of the probability p_{AFB} to have an antiferromagnetic bond is presented in the interval $p_{AFB} \in [0.09, 0.13]$. According to Dennis et al. [2], $p_{AFB,crit}$ is expected to be located in this region.

Then in Subsection 5.2, the interval $p_{AFB} \in [0.090, 0.120]$, where the phase crossing was observed in the first series of runs, is analysed in more detail.

Thereafter, the results for $p_{AFB,crit}$ obtained in Subsection 5.2 are analysed and discussed in Subsection 5.3. They are then compared to literature values in Subsection 5.4.

5.1. Overview

The first series of runs of the simulation of the RBIM was performed to coarsely check if the phase crossing actually occurs where it is expected according to Dennis et al. [2], namely around the value of $p_{AFB,crit} = 0.1094 \pm 0.0002$. Therefore, the values of the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class were determined in the interval of the probability p_{AFB} to have an antiferromagnetic bond ranging from $p_{AFB} = 0.09$ to $p_{AFB} = 0.13$. This was done with both variants of the algorithm, the one that uses the WHO- and the other that uses the WHAT-worm algorithm. The results of the WHO-variant are shown in Figure 5.1.1, those of the WHAT-variant in Figure 5.1.2.

Both variants of the algorithm produce a similar outcome. The differences between the outcomes of the two variants will be discussed in detail in Section 6. For both variants, $P_{trivial}$ starts at almost 1 for $p_{AFB} = 0.09$, indicating that the phase crossing starts slightly before that value of p_{AFB} . Thereafter, for both variants, $P_{trivial}$ decreases drastically as p_{AFB} gets larger. Finally, they both reach the other phase at $p_{AFB} = 0.13$, where their values for $P_{trivial}$ are around 0.25. This shows that the phase crossing occurs in this region of p_{AFB} and that thus the critical point of the phase transition has to lie herein.

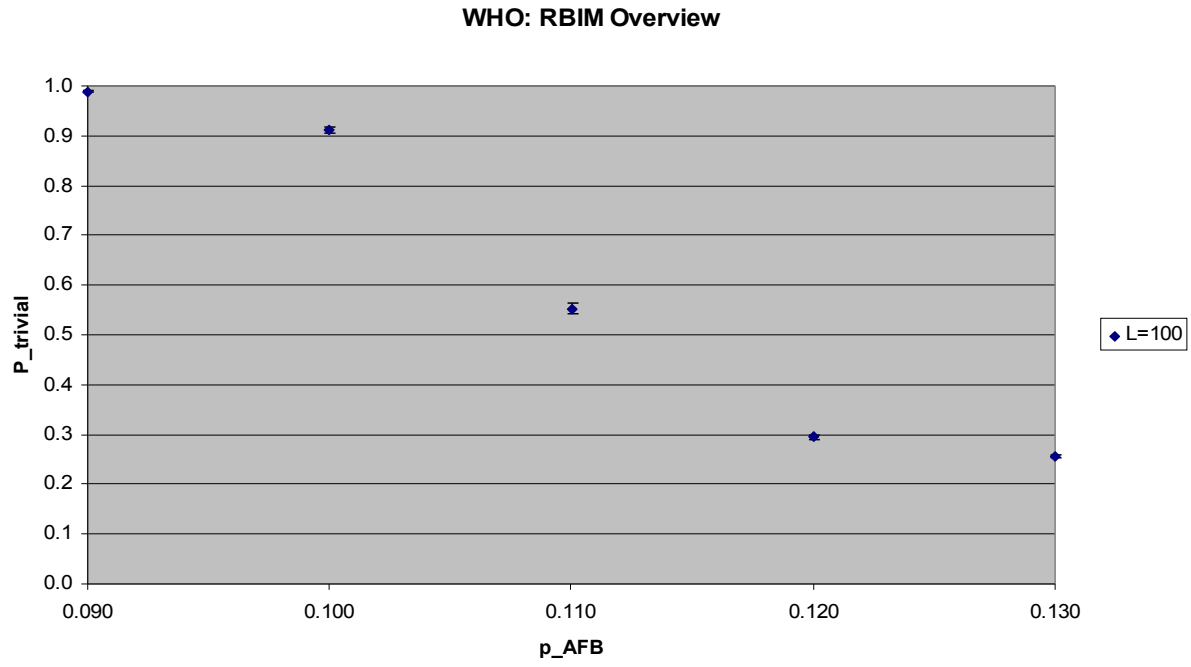

Figure 5.1.1: Diagram illustrating the results of the numerical calculations of the "overview" series of runs performed using the "wormhead only" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability p_{AFB} to have an antiferromagnetic bond. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. The data set was created using a square lattice with a side length of 100 links.
These results give a coarse overview on how $P_{trivial}$ behaves as a function of p_{AFB} in the interval where the phase crossing is expected according to Dennis et al. [2].

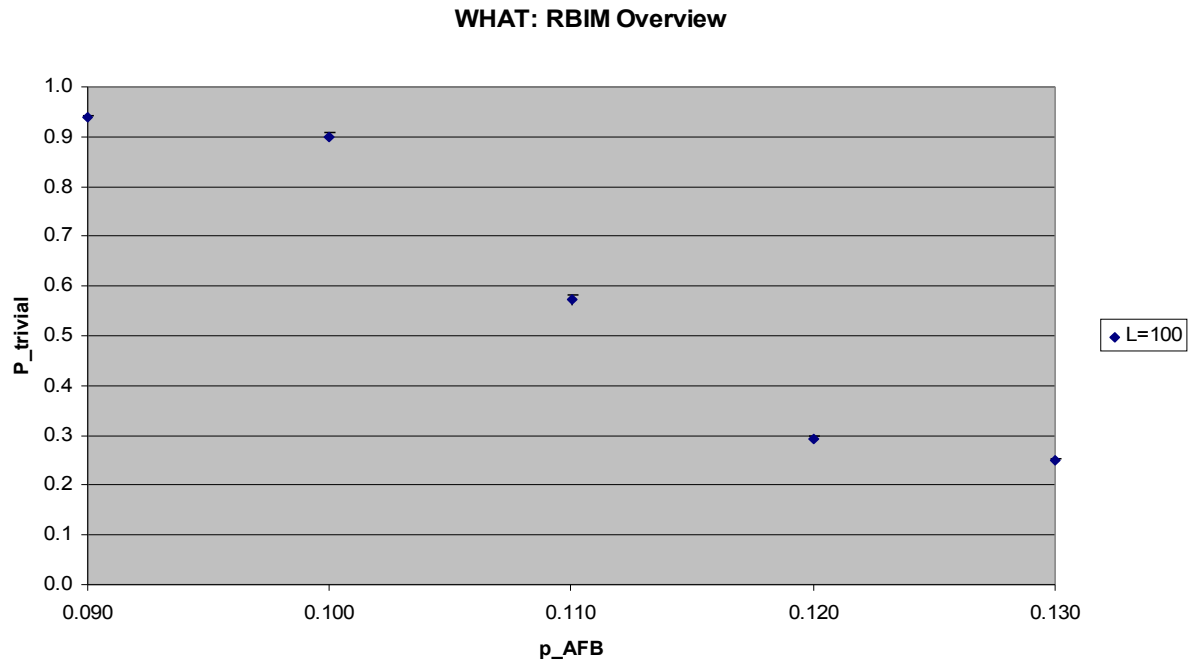

Figure 5.1.2: Diagram illustrating the results of the numerical calculations of the "overview" series of runs performed using the "wormhead and -tail" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability p_{AFB} to have an antiferromagnetic bond. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. The data set was created using a square lattice with a side length of 100 links.
These results give a coarse overview on how $P_{trivial}$ behaves as a function of p_{AFB} in the interval where the phase crossing is expected according to Dennis et al. [2].

5.2. Zoom-in on interval around $p_{AFB,crit}$

In the second series of runs from the simulation of the RBIM, a detailed analysis of the behaviour of the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class as a function of the probability p_{AFB} to have an antiferromagnetic bond was made in the interval ranging from $p_{AFB}=0.090$ to $p_{AFB}=0.120$, the interval where the location of the phase transition is expected according to Dennis et al. [2] and where the phase crossing was observed in the first series of runs. In contrast to the previous series of runs, this analysis determined the value of $P_{trivial}$ with a higher resolution. The distance between two adjacent p_{AFB}, for which $P_{trivial}$ was determined, was $\Delta p_{AFB}=0.001$. The accuracy of these $P_{trivial}$ determinations was kept as high as in the previous series of runs with 10^7 homology class measurements at each p_{AFB}. This was again done with both variants of the algorithm. The results of the WHO-variant are shown in Figure 5.2.1, those of the WHAT-variant in Figure 5.2.2.

Both variants of the algorithm produce very similar results. These are compared in detail in Section 6. They show the same phase crossing behaviour that was already observed in the results of the previous series of runs. This time, however, in a better resolution with regard to p_{AFB} and for different lattice sizes. It can be seen how for the data sets of each lattice size, the crossing starts in the first phase for $p_{AFB}=0.090$, where the corresponding value of $P_{trivial}$ is close to 1. Then for all of them, $P_{trivial}$ decreases as p_{AFB} gets larger. And finally, for all of them $P_{trivial}$ gets close to 0.25, when p_{AFB} becomes 0.120. In addition to that, it can be observed that with increasing lattice size the plateaux of the phases extend farther. As a consequence, the phase crossing has to occur in a smaller interval. This causes the value of $P_{trivial}$ to drop steeper during the phase crossing for increasing lattice sizes. Furthermore, it can be seen that at the point $p_{AFB}=0.109\pm0.001$ (WHO) respectively $p_{AFB}=0.108\pm0.001$ (WHAT) the graphs from the different lattice sizes intersect each other.

The analysis of these results is done sophisticatedly and thus performed separately in Subsection 5.3.

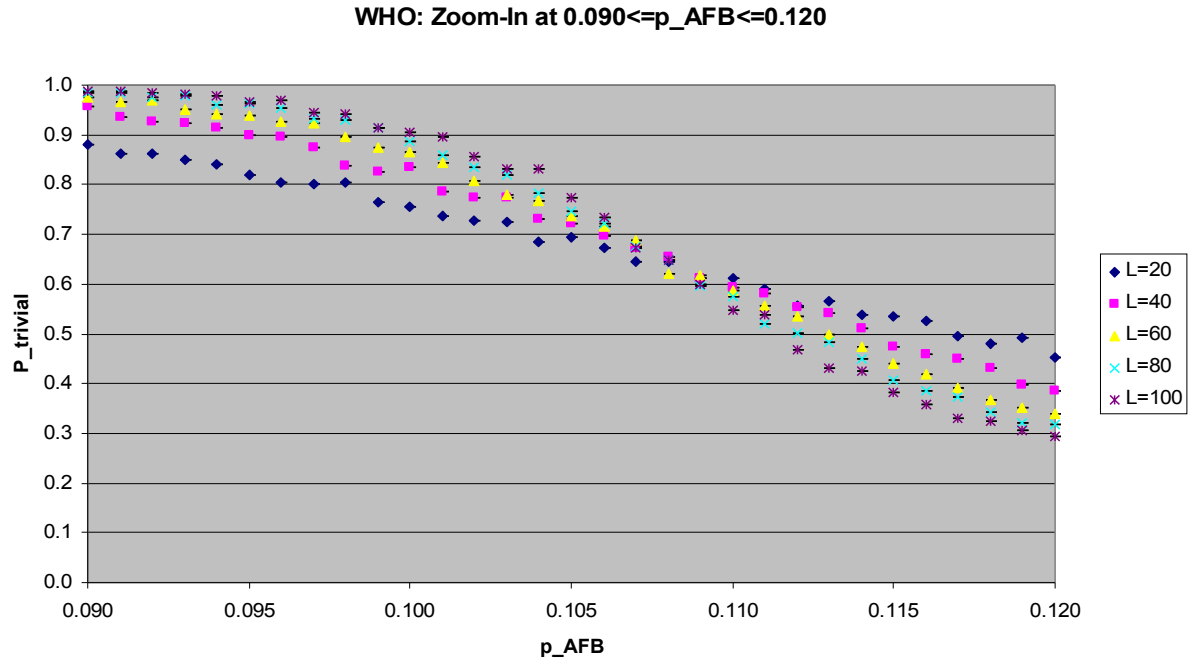

Figure 5.2.1: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.090<=q<=0.120" series of runs performed using the "wormhead only" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability p_{AFB} to have an antiferromagnetic bond. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of p_{AFB} in the interval between $p_{AFB}=0.090$ and $p_{AFB}=0.120$. Note the small size of the error bars due to the high accuracy of the results due to the large number of worm algorithm applications.

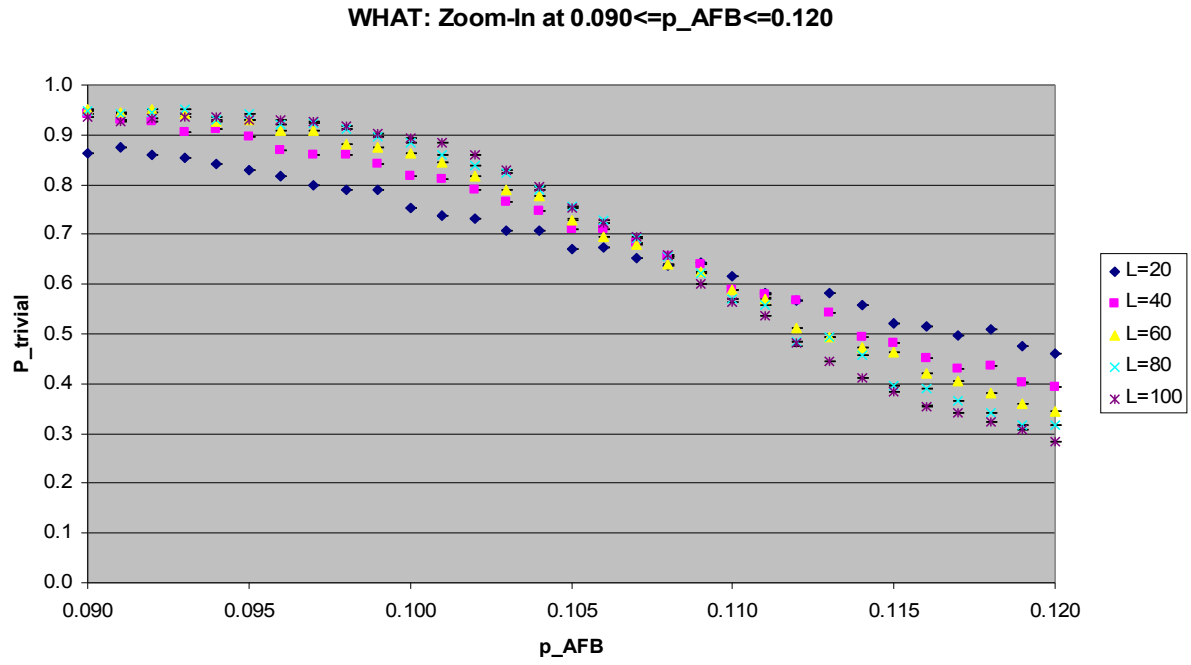

Figure 5.2.2: Diagram illustrating the results of the numerical calculations of the "zoom-in at 0.090<=q<=0.120" series of runs performed using the "wormhead and -tail" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability p_{AFB} to have an antiferromagnetic bond. The vertical axis represents the proportion $P_{trivial}$ of produced configurations that belong to the trivial homology class. Five different data sets were created. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links.
These results show how $P_{trivial}$ behaves as a function of p_{AFB} in the interval between $p_{AFB}=0.090$ and $p_{AFB}=0.120$. Note the small size of the error bars due to the high accuracy of the results due to the large number of worm algorithm applications.

5.3. Analysis of the results

The "zoom-in at 0.090<=p_AFB<=0.120" series of runs faced the same problem in regard to the method of using the start and the end of the phase crossing to determine the interval in which $p_{AFB,crit}$ is located as the "zoom-in at 0.405<=q<=0.420" series of runs in the simulation of the SIM, the phase crossing starts and ends approximately at the same positions as in the previous series of runs and thus the interval keeps its length. Therefore, also here an alternative approach needed to be used. Since the performance of this series of runs was already very time consuming, the simulation on a lattice of significantly larger size to narrow down the width of the phase crossing was not an option, as this would have increased the calculation time into an unrealisable range. Hence, the same method as for the analysis of the "zoom-in at 0.405<=q<=0.420" series of runs in the simulation of the SIM was used:
The direct view of these results reveals that with increasing lattice size, the graphs of $P_{trivial}$ become steeper, that these graphs intersect near $p_{AFB}=0.109\pm0.001$ (WHO) respectively $p_{AFB}=0.108\pm0.001$ (WHAT) and that in the vicinity of this intersection point these graphs behave almost linearly. Thus all the requirements are fulfilled to draw conclusions on the critical value of p_{AFB} by analysing the intersections of the linear regressions of these graphs in this vicinity. These linear regressions are shown in Figure 5.3.1 for the WHO- and Figure 5.3.2 for the WHAT-variant of the algorithm. Each of the regression lines intersects with the other ones at distinct points, no intersection points overlap, as it can be seen in Table 5.3.1, where the positions of these intersections are presented. Hence, it is assumed that these intersection points stochastically distribute around $p_{AFB,crit}$. Therefore, their mean value can be used to determine $p_{AFB,crit}$, together with their standard deviation for its error. For the WHO-variant of the algorithm, these are

$$p_{AFB,crit,WHO}=0.1081\pm0.0007 \tag{5.3.1}$$

and for the WHAT-variant

$$p_{AFB,crit,WHAT}=0.108\pm0.001. \tag{5.3.2}$$

Analogously, the value of $P_{trivial}$ at $p_{AFB,crit}$ and its error can be determined. For the WHO-variant of the algorithm, these are

$$P_{trivial,WHO}\left(p_{AFB,crit}\right)=0.642\pm0.017 \tag{5.3.3}$$

and for the WHAT-variant

$$P_{trivial,WHAT}\left(p_{AFB,crit}\right)=0.652\pm0.036. \tag{5.3.4}$$

The two values obtained for $p_{AFB,crit}$ from the two different variants of the algorithm, which correspond to the accuracy threshold p_{crit}, lie apart less than 1σ with respect to the errors of both values. The same is true for their $P_{trivial}\left(p_{AFB,crit}\right)$ and the corresponding errors. This indicates that the deviation between these values is of a stochastic nature and that thus for this level of accuracy both variants of the algorithm produce the same results. This corresponds to the findings from Section 6 that, although a systematic deviation was observed between the two variants of the algorithm, in this series of runs the distribution of the differences Δ of the determined values of $P_{trivial}$ from the two variants of the algorithm at each data point do not differ that much from the normal distribution as in other series of runs. The systematic deviation needs a higher level of accuracy to have an effect.

Intersection	WHO		WHAT	
L_1;L_2	p_AFB	P_trivial	p_AFB	P_trivial
20;40	0.10960	0.61080	0.10958	0.61734
20;60	0.10871	0.62514	0.10842	0.63382
20;80	0.10833	0.63110	0.10859	0.63129
20;100	0.10826	0.63231	0.10816	0.63742
40;60	0.10756	0.65857	0.10696	0.67747
40;80	0.10754	0.65898	0.10788	0.65629
40;100	0.10773	0.65453	0.10737	0.66801
60;80	0.10752	0.65957	0.10915	0.61192
60;100	0.10781	0.65131	0.10770	0.65519
80;100	0.10805	0.64102	0.10565	0.73427

Table 5.3.1: Positions of the intersections between the linear regression graphs of the data points around $p_{AFB}=0.109$ *(WHO) respectively* $p_{AFB}=0.108$ *(WHAT) in the "zoom-in at* 0.090<=p_AFB<=0.120*" series of runs from the simulation of the RBIM. The left column indicates the different intersections. The two numbers name the lattice sizes of which the linear regression graphs intersect. This occurs at the coordinates presented in the central and right column doublet for the WHO- respectively WHAT-variant of the algorithm.*

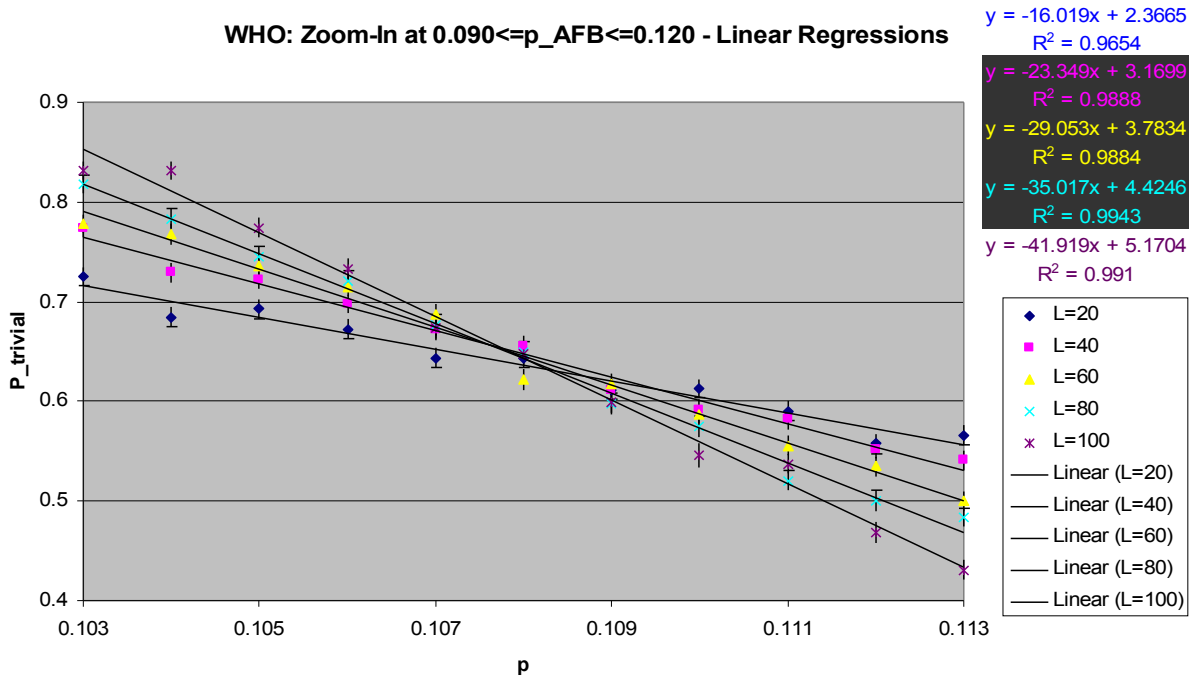

Figure 5.3.1: Diagram showing the linear regression on the result data sets of the numerical calculations of the "zoom-in at 0.090<=p_AFB<=0.120*" series of runs performed using the "wormhead only" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability* p_{AFB} *to have an antiferromagnetic bond. The vertical axis represents the proportion* $P_{trivial}$ *of produced configurations that belong to the trivial homology class. Five different data sets are shown. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links. The equations of the regression graphs and their coefficients of determination are listed on the top right. The colours of the equations match those of the individual data sets. Bright colours have a dark background for better readability.*
For the regression only the data points between $p_{AFB}=0.103$ *and* $p_{AFB}=0.113$ *were used, because here the behaviour of* $P_{trivial}(p_{AFB})$ *was almost linear.*

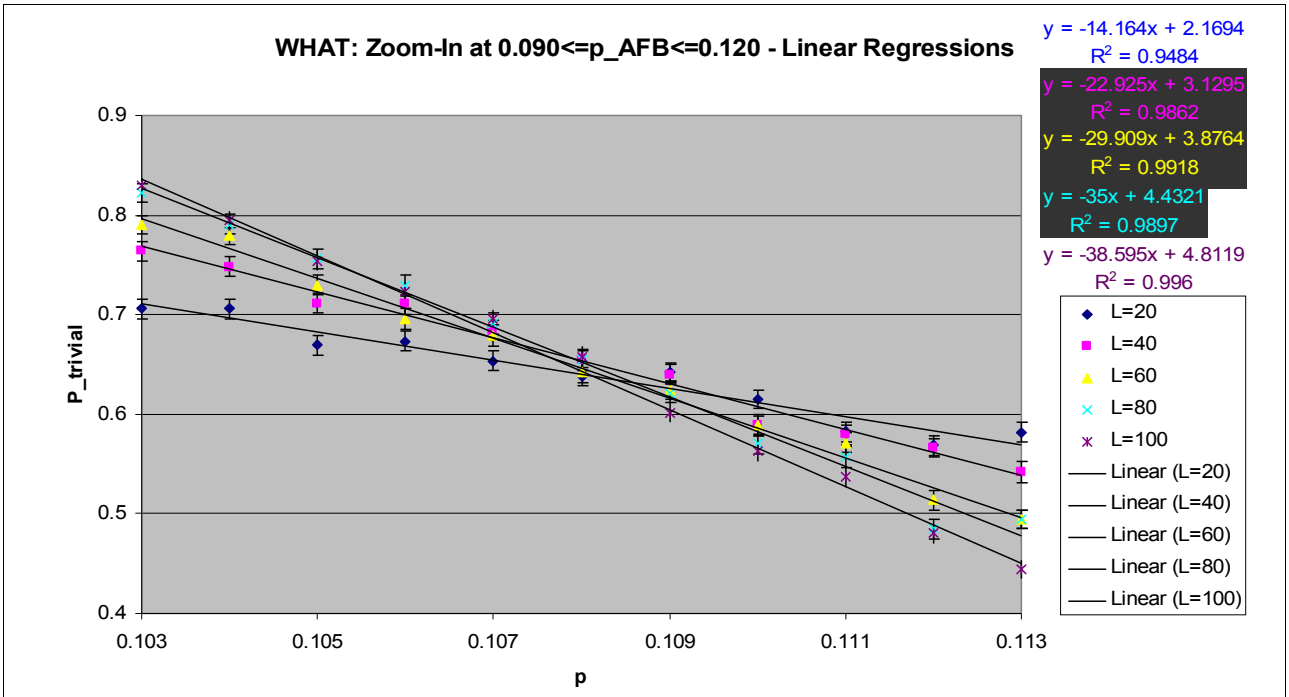

Figure 5.3.2: Diagram showing the linear regression on the result data sets of the numerical calculations of the "zoom-in at 0.090<=p_AFB<=0.120*" series of runs performed using the "wormhead and -tail" variant of the algorithm in the simulation of the RBIM. The horizontal axis represents the probability* p_{AFB} *to have an antiferromagnetic bond. The vertical axis represents the proportion* $P_{trivial}$ *of produced configurations that belong to the trivial homology class. Five different data sets are shown. Each one used a square lattice of a different size, namely a side length of 20 (dark blue), 40 (pink), 60 (yellow), 80 (light blue), respectively 100 (purple) links. The equations of the regression graphs and their coefficients of determination are listed on the top right. The colours of the equations match those of the individual data sets. Bright colours have a dark background for better readability.*
For the regression only the data points between $p_{AFB}=0.103$ *and* $p_{AFB}=0.113$ *were used, because here the behaviour of* $P_{trivial}(p_{AFB})$ *was almost linear.*

5.4. Comparison to other reference sources

For the toric code error correction, the position of the accuracy threshold p_{crit} has been determined by Dennis et al. [2]. They describe the analogy between the toric code error correction and the RBIM and show that the critical value $p_{AFB,crit}$ of the probability to have an antiferromagnetic bond in the RBIM along the Nishimori line corresponds to the accuracy threshold p_{crit}. Then they cite Honecker et al. [5], who have simulated the RBIM and there determined the Nishimori point N using the numerical method known as *finite size scaling*. According to them, it lies at

$$p_{crit,[5]}=0.1094\pm0.0002\,. \tag{5.4.1}$$

This value lies $1.88\,\sigma$ (WHO) respectively $1.27\,\sigma$ (WHAT) away from the accuracy threshold determined in this thesis. There are several possible causes for this deviation:
One possible cause is the systematic deviation between the two variants of the worm algorithm. Although it did not cause a significant difference between the outcome of the distinct algorithms in this series of runs, it still indicates that they do not work properly and cause an error, which is of the same order of magnitude as the deviation observed here. The idea that the worm algorithms do not work properly is also supported by the fact that they did not produce the correct results for q at the periphery of the interval $q\in[0,1]$ during the tests.
The most plausible cause, however, is the missing thermalization phase at the beginning of the algorithm. Due to its lack, the first data points that were determined are autocorrelated. Since,

however, all the data points need to be stochastically distributed, this will cause a bias. Although this bias will decrease with increasing size of the set of data points, it might here still have the small effect that is observed as this deviation.
Due to the limited scope of this thesis, it was not possible to determine which of these error sources actually causes this deviation. It is, however, plausible to assume that they all contribute to it to some degree. Thus, in order to improve the method for determination of the accuracy threshold p_{crit} presented in this thesis, they all should be eliminated by modifying the algorithm accordingly. While it is still unclear what exactly causes the systematic deviation between the two variants of the worm algorithm and hence what exactly needs to be changed, the implementation of a thermalization phase is easily realized by letting the worm algorithms initially create configurations without evaluating their homology class. Based on the thermalization test data presented in Section 3.3, omitting the first $2 \cdot 10^6$ or slightly more configurations should be enough to get rid of the autocorrelation effects.

6. Comparison of the worm algorithm performance

In order to test for the consistency of the produced results of the numerical simulations, they were performed twice, using algorithms that applied two different variants of the worm algorithm, namely the *wormhead only* (WHO) and the *wormhead and -tail* (WHAT) variant. If both worm algorithms work properly, they should yield the same results, up to a stochastic deviation. However, if this is not the case and some systematic deviation is observed, it would indicate that the worm algorithms still contain a methodological or a coding error, which needs to be corrected.
In order to systematically compare the two different variants of the worm algorithm, for every data point (calculation performed at a distinct input parameter q respectively p_{AFB}) of every data set (different lattice sizes) of every series of runs of each of the two simulations the difference Δ between the results obtained using the two variants of the worm algorithm was determined. To each of these results an error s_i is associated. According to the laws of error propagation, the error of the difference s_Δ is thus given by

$$s_\Delta = \sqrt{{s_{WHO}}^2 + {s_{WHAT}}^2} \tag{6.1}$$

Since both algorithms should produce the same result, their difference should always be 0. Due to the probabilistic nature of the used method, the obtained values for this difference should stochastically distribute around this value with a standard deviation of $\sigma = s_\Delta$. Hence, the values of the ratio $\frac{\Delta}{s_\Delta}$ from the different data sets should distribute normally like shown in Table 6.1.

Δ/s_Δ	0-1	1-2	2-3	>3
Distribution	0.6827	0.2718	0.0428	0.0027

Table 6.1: Expected normal distribution around 0 of the values of the difference Δ between the results of the two variants of the algorithm. Δ / s_Δ is the value of the deviation from 0 normalized by the corresponding standard deviation. The spectrum of values it can obtain is divided here in four intervals for reasons of clarity. The distribution below indicates how the deviations should distribute themselves among these intervals, if their distribution is normal.

The results this analysis yields for the different series of runs are depicted in Table 6.2. While in the "Overview" series of runs of the SIM simulation the values of this difference abide well to the normal distribution, this is no longer the case for the other series of runs, where the aberration from the normal distribution is either small (like in the "Zoom-In at 0.220<=q<=0.268" and "Zoom-In at 0.350<=q<=0.450" series of runs of the SIM simulation) or extreme (like in the "Zoom-In at 0.405<=q<=0.420" series of runs of the SIM simulation). A closer look reveals that the aberration gets stronger the more accurate (in terms of number of worm algorithm applications per data point) the calculation was performed. This indicates that there is a systematic error source causing a small but non-vanishing difference between the results of the different worm algorithm variants. This error was not found within the time available for completing this thesis. However, since this difference is small, its effect on the obtained results is small as well. They can thus still be used till a certain degree of accuracy and it has to be kept in mind that the actual error estimates should be taken with a grain of salt.

SIM: Overview

Δ/s_Δ	0-1	1-2	2-3	>3
L = 20	0.6634	0.3069	0.0198	0.0099
L = 40	0.6832	0.2673	0.0396	0.0099
L = 60	0.7030	0.2574	0.0198	0.0198
L = 80	0.6931	0.2376	0.0495	0.0198
L = 100	0.6535	0.3069	0.0297	0.0099

SIM: Zoom-In at 0.220<=q<=0.268

Δ/s_Δ	0-1	1-2	2-3	>3
L = 20	0.6939	0.3061	0.0000	0.0000
L = 40	0.6122	0.3878	0.0000	0.0000
L = 60	0.5918	0.4082	0.0000	0.0000
L = 80	0.6122	0.3878	0.0000	0.0000
L = 100	0.6327	0.3673	0.0000	0.0000

SIM: Zoom-In at 0.350<=q<=0.450

Δ/s_Δ	0-1	1-2	2-3	>3
L = 20	0.5941	0.3168	0.0891	0.0000
L = 40	0.6634	0.2772	0.0594	0.0000
L = 60	0.5842	0.3465	0.0495	0.0198
L = 80	0.5644	0.3663	0.0495	0.0198
L = 100	0.6436	0.3366	0.0198	0.0000

SIM: Zoom-In at 0.405<=q<=0.420

Δ/s_Δ	0-1	1-2	2-3	>3
L = 20	0.0625	0.3125	0.5625	0.0625
L = 40	0.0000	0.0000	0.0000	1.0000
L = 60	0.0000	0.0000	0.0000	1.0000
L = 80	0.0000	0.0000	0.0000	1.0000
L = 100	0.0000	0.0000	0.0625	0.9375

SIM: Zoom-In at 0.4130<=q<=0.4150

Δ/s_Δ	0-1	1-2	2-3	>3
L = 1000	0.4762	0.0952	0.1429	0.2857

RBIM: Overview

Δ/s_Δ	0-1	1-2	2-3	>3
L = 100	0.2000	0.6000	0.0000	0.2000

RBIM: Zoom-In at 0.090<=p_AFB<=0.120

Δ/s_Δ	0-1	1-2	2-3	>3
L = 20	0.5806	0.3548	0.0645	0.0000
L = 40	0.5484	0.3226	0.1290	0.0000
L = 60	0.4839	0.3548	0.0645	0.0968
L = 80	0.5484	0.1613	0.0645	0.2258

Table 6.2: Distribution of the difference Δ between the results of the two variants of the algorithm normalized by the corresponding error s_Δ for every data set (lattice size) of every series of runs. The distribution distinguishes if the value of Δ / s_Δ falls into one of four different intervals. The expected normal distribution is presented in Table 6.1.
As it can be seen, only the "SIM: Overview" distribution matches that expectation. The other distributions deviate from the expectation with variable severity.

7. Further findings

This section presents the findings that were made during this thesis in addition to those related to the determination of the accuracy threshold p_{crit} :

When looking at the series of runs that contain more than one data set, i.e. were performed on more than one lattice size, it can be seen that the behaviour of $P_{trivial}$ becomes steeper during the phase crossing with increasing lattice size and occurs over a shorter interval of the link switch probability q respectively probability p_{AFB} to have an antiferromagnetic bond. This behaviour corresponds to expectation, since in the infinite-size lattice model the phase transition occurs at one point, namely q_{crit} respectively $p_{AFB,crit}$, and the lattices of finite size approximate this behaviour more accurately the larger they become. Furthermore, for all probabilities q respectively p_{AFB} that have smaller values than the start of the crossover, the value of $P_{trivial}$ is approximately 1. For all probabilities q respectively p_{AFB} that have larger values than the end of the crossover, the value of $P_{trivial}$ is approximately 0.25. This characterizes nicely the different nature of the two phases in the two-dimensional Ising model. The differences to the strict values of 1 respectively 0.25 in the infinite-size lattice model can be explained by the finiteness of the lattice sizes that were used, which allows only the approximation of the infinite-size behaviour, as well as the approximative nature of the here used method. To summarize, these findings indicate that the here used algorithms perform well in simulating the behavioural features of the Ising model respectively the toric code error correction model.

Since on the infinite-size lattice at the critical point q respectively p_{crit} the value of $P_{trivial}$ instantly drops from 1 to 0.25, its behaviour when passing the phase boundary is discontinuous. Therefore, this parameter behaves as if the phase transition were of the first order, while actually the phase transition in the two dimensional Ising model is known to be of the second order.

As derived in Section 3.1, the critical point in the standard Ising model lies at

$$q_{crit} = \sqrt{2} - 1 \approx 0.41421... \quad . \tag{7.1}$$

For a lattice of infinite size, at this point the phase transition occurs. During this thesis, it was, however, serendipitously discovered that for all tested lattices out of a large variety of small sizes, $P_{trivial}$ had always the same value at this point. The calculations were performed using the brute force algorithm so that the determined values are accurate. They always yielded

$$P_{trivial}\left(q_{crit}\right) = 0.5 \tag{7.2}$$

as it can also be seen in the individual results presented in Table 7.1 and Table 7.2.

L_hor	L_ver	P_trivial	P_vertical	P_horizontal	P_both
1	1	0.50000	0.20711	0.20711	0.08579
2	2	0.50000	0.20000	0.20000	0.10000
3	3	0.50000	0.19351	0.19351	0.11299
4	4	0.50000	0.19034	0.19034	0.11933
1	2	0.50000	0.08579	0.35355	0.06066
1	3	0.50000	0.03553	0.43365	0.03082
1	4	0.50000	0.01472	0.47141	0.01388
1	5	0.50000	0.00610	0.48795	0.00595
1	6	0.50000	0.00253	0.49498	0.00250

Table 7.1: First part of the results from the brute force analysis of $P_{trivial}\left(q_{crit}\right)$ in the standard Ising model. The two columns on the left show the different lattice sizes. The four columns on the right show the proportions of the sums of weights of configurations belonging to a specific homology class for these lattice sizes at $q = q_{crit} = \sqrt{2} - 1$. These values have been determined using the brute force algorithm and can thus be considered accurate up to rounding. The data sets in the first four rows are from square lattices, thereafter the data sets from non-square lattices are shown.

L_hor	L_ver	P_trivial	P_vertical	P_horizontal	P_both
1	7	0.50000	0.00105	0.49791	0.00104
1	8	0.50000	0.00043	0.49913	0.00043
1	9	0.50000	0.00018	0.49964	0.00018
1	10	0.50000	7.434E-5	0.49985	7.432E-5
1	11	0.50000	3.079E-5	0.49994	3.079E-5
1	12	0.50000	1.275E-5	0.49997	1.275E-5
1	13	0.50000	5.283E-6	0.49999	5.283E-6
1	14	0.50000	2.188E-6	0.50000	2.188E-6
1	15	0.50000	9.064E-7	0.50000	9.064E-7
1	16	0.50000	3.755E-7	0.50000	3.755E-7
1	17	0.50000	1.555E-7	0.50000	1.555E-7
1	18	0.50000	6.442E-8	0.50000	6.442E-8
1	19	0.50000	2.668E-8	0.50000	2.668E-8
2	1	0.50000	0.35355	0.08579	0.06066
2	3	0.50000	0.12203	0.28619	0.09179
2	4	0.50000	0.07895	0.35088	0.07018
2	5	0.50000	0.05212	0.39824	0.04964
2	6	0.50000	0.03448	0.43172	0.03379
2	7	0.50000	0.02272	0.45474	0.02253
2	8	0.50000	0.01490	0.47025	0.01485
2	9	0.50000	0.00973	0.48055	0.00972
3	1	0.50000	0.43365	0.03553	0.03082
3	2	0.50000	0.28619	0.12203	0.09179
3	4	0.50000	0.14074	0.25114	0.10813
3	5	0.50000	0.10672	0.29968	0.09360
4	1	0.50000	0.47141	0.01472	0.01388
4	2	0.50000	0.35088	0.07895	0.07018
4	3	0.50000	0.25114	0.14074	0.10813
5	1	0.50000	0.48795	0.00610	0.00595
5	2	0.50000	0.39824	0.05212	0.04964
5	3	0.50000	0.29968	0.10672	0.09360
6	1	0.50000	0.49498	0.00253	0.00250
6	2	0.50000	0.43172	0.03448	0.03379
6	3	0.50000	0.34048	0.08238	0.07714
7	1	0.50000	0.49791	0.00105	0.00104
7	2	0.50000	0.45474	0.02272	0.02253
8	1	0.50000	0.49913	0.00043	0.00043
8	2	0.50000	0.47025	0.01490	0.01485
9	1	0.50000	0.49964	0.00018	0.00018
9	2	0.50000	0.48055	0.00973	0.00972
10	1	0.50000	0.49985	7.434E-5	7.432E-5
11	1	0.50000	0.49994	3.079E-5	3.079E-5
12	1	0.50000	0.49997	1.275E-5	1.275E-5
13	1	0.50000	0.49999	5.283E-6	5.283E-6
14	1	0.50000	0.50000	2.188E-6	2.188E-6
15	1	0.50000	0.50000	9.064E-7	9.064E-7
16	1	0.50000	0.50000	3.755E-7	3.755E-7

Table 7.2: Second part of the results from the brute force analysis of $P_{trivial}(q_{crit})$ in the standard Ising model. The two columns on the left show the different lattice sizes. The four columns on the right show the proportions of the sums of weights of configurations belonging to a specific homology class for these lattice sizes at $q = q_{crit} = \sqrt{2} - 1$. These values have been determined using the brute force algorithm and can thus be considered accurate up to rounding. The data sets in the rows continue to be from non-square lattices.

8. Conclusions

The two variants of the algorithm used to determine the accuracy threshold p_{crit} based on the analogy to the random-bond Ising model yielded the following results:

$$p_{crit,WHO} = 0.1081 \pm 0.0007 \tag{8.1}$$

respectively

$$p_{crit,WHAT} = 0.108 \pm 0.001 \tag{8.2}$$

These values are close to the reference value provided by Dennis et al. [2]. The deviation between them lies between 1σ and 2σ, so the obtained results are not significantly different from the reference value. However, that the deviation is larger than 1σ might indicate that the method, which is applied in this thesis, produces a systematic error that slightly biases the results. Possible sources for this error are the improper functioning of the worm algorithms (as it is also indicated by the deviation between the results, which the two variants produce for the same input parameters) and the omission of a thermalization phase. How much these different sources contribute to the bias and whether there are further sources has not yet been determined and shall be the subject of future research.

9. Acknowledgements

I would like to thank all the persons who helped me during this Master Thesis.

Especially I want to thank my supervisors Prof. Dr. Uwe-Jens Wiese and Dr. David Mesterhazy for their support, as well as Stephan Caspar for his help in using the ITP PC-cluster.

10. References

[1]: M. A. Nielsen, I. L. Chuang; Quantum Computation and Quantum Information; 10th Anniversary Edition; Cambridge University Press; 2017 (reprint); p. 13-28
[2]: E. Dennis et al.; Topological quantum memory; Journal of Mathematical Physics; 2002; Volume 43; p. 4452-4505
[3]: U. Wenger; Statistische Thermodynamik I, Lecture Notes, Spring Semester 2015
[4]: L. Onsager; Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition; Physical Review; 1944; Volume 65; p. 117-149
[5]: A. Honecker, M. Picco, P. Pujol; Nishimori point in the 2D±J random-bond Ising model; Physical Review Letters; 2001; Volume 87:047201

Erklärung

gemäss Art. 30 RSL Phil.-nat. 18

Name/Vorname: Zbinden, Matti

Matrikelnummer: 12-103-883

Studiengang: Theoretische Physik

Bachelor ☐ Master ☑ Dissertation ☐

Titel der Arbeit: Quantum Error Correction for the Toric Code

LeiterIn der Arbeit: Prof. Uwe-Jens Wiese

Ich erkläre hiermit, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe r des Gesetzes vom 5. September 1996 über die Universität zum Entzug des auf Grund dieser Arbeit verliehenen Titels berechtigt ist.
Für die Zwecke der Begutachtung und der Überprüfung der Einhaltung der Selbständigkeitserklärung bzw. der Reglemente betreffend Plagiate erteile ich der Universität Bern das Recht, die dazu erforderlichen Personendaten zu bearbeiten und Nutzungshandlungen vorzunehmen, insbesondere die schriftliche Arbeit zu vervielfältigen und dauerhaft in einer Datenbank zu speichern sowie diese zur Überprüfung von Arbeiten Dritter zu verwenden oder hierzu zur Verfügung zu stellen.

Bern, 12.02.2019

Ort/Datum

Unterschrift M. Zbinden